

IEEE Signal Processing MAGAZINE

Volume 36 | Number 6 | November 2019



BRAIN-INSPIRED COMPUTING

Algorithmic, Hardware, and Neuroscience Perspectives

Speech Processing for Digital Home Assistants

Medicine on the Move: Wearable Devices

Monitoring Active Volcanoes

IEEE
Signal
Processing
Society



Call for Papers IEEE Signal Processing Magazine

Special Issue: Innovation Starts with Education

Signal processing (SP) is at the very heart of our digital lives, owing to its role as the pivotal development technology across multiple disciplines. Its prominence in modern data science has created a necessity to supply industry, government labs, and academia with graduates who possess relevant crucial signal processing expertise and are well equipped to deal with the manifold challenges in current and future applications. To this end, both the ways to deliver the educational content and the core Signal Processing curriculum need to be revisited and integrated into Engineering and Computer Science degrees, to provide hands-on skills, experience, and inspiration for students.

Signal processing education in today's universities is largely influenced by three modern trends: 1) the availability of competing and complementary online and multimedia resources; 2) the fact that we live in a world in which the amount and diversity of information we generate, process and analyse is growing ever faster; 3) the explosive growth of computing power, and the rapid development of new technologies for implementing both analog and digital signal processing. These trends offer both opportunities and challenges, which we can, and must, exploit in charting dynamically adjustable courses that attract a high level of student engagement while offering a mix of essential background physics, intuition, mathematical rigor, and practical applicability of the taught material.

With such initiatives underway at many universities world-wide, this special issue aims to facilitate both keeping abreast with SP education and exploring innovative and participatory ways to present the educational materials. In effect, we cannot assume that students will be able to appreciate the scope and relevance of their courses without explicitly building a bridge between the material presented in class and cutting-edge research, societal, and practical impact of their education. This includes the convergence of educational material with other disciplines (machine learning, data science, big data, bioengineering, artificial intelligence, finance and many others).

This special issue will revolve around three general and most pressing aspects of modern SP education:

- **How to educate differently (better).** This includes the use of available technology, bringing research into the classroom, web resources, experiential learning, and massive open online courses (MOOC).
- **Student engagement,** such as ways to enhance student creativity and curiosity, student satisfaction issues, various forms of assessment and metrics, engagement of under-represented population, and outreach drives.
- **Promotion of the societal impact of SP,** including privacy, ethical and security concerns, wearable devices and eHealth, global interconnections through IoT, and impact on climate change, global economy and finance.

Topics of interest include but are not limited to:

- Mitigation of issues related to the perceived difficulty of traditional SP courses, such as strategies on how to teach SP with less maths and how to attract attendees from non-engineering departments.
- The use of emerging technologies and technologically orientated classroom, such as MOOC and web resources.
- Metrics for success of education delivery in the After Online Technology (AOT) era.
- Using the principles of signal processing to improve teaching and research in related areas, such as machine learning, bioengineering, artificial intelligence and optimization, and vice versa.
- Curricular changes to meet contemporary demands from industry, such as using practically relevant problems, exploring feasible extensions and new applications of the taught material, and curiosity driven learning.
- Preparing students for life-long learning, teaching life-long fundamentals of SP, and relevance of SP with respect to technological advance.
- Challenges and solutions in industry-run courses; design of short courses offered by academia for industry, Government Agencies and National Defence.
- Role of mentorship and initiatives to encourage and motivate students in research experiences.
- Promoting creativity in learning, especially when applying the concepts with "opportunity windows" to explore entrepreneurship and possible product developments, and cross-disciplinary aspects of our work.

Submission Process. Original submissions will be reviewed according to the guidelines as set out in the IEEE Signal Processing Magazine, and should be submitted online at <http://mc.manuscriptcentral.com/sps-ieee>. Prospective authors should initially submit short White Papers, as indicated below, which will be reviewed by Guest Editors. Authors of successful White Papers will be invited to submit full-length manuscripts, which will undergo the usual reviewing process in accordance with the schedule outlined below.

White Papers (up to 4 pages) due: February 1, 2020

Full length manuscripts due: May 1, 2020

Revised manuscripts due: September 1, 2020

Final manuscripts due: December 1, 2020

Decision on White Papers: March 1, 2020

Review results and decision notification: July 1, 2020

Acceptance notification: November 1, 2020

Publication date: March 1, 2021

Guest Editors

- Mónica Bugallo, SUNY at Stony Brook, USA, monica.bugallo@stonybrook.edu
- Anthony Constantinides, Imperial College London, a.constantinides@imperial.ac.uk
- Danilo Mandic, Imperial College London, UK, d.mandic@imperial.ac.uk
- Alan Oppenheim, MIT, USA, avo@mit.edu
- Roberto Togneri, University of Western Australia, roberto.togneri@uwa.edu.au

Please contact Danilo Mandic for any initial questions concerning this Special Issue.

Contents

Volume 36 | Number 6 | November 2019

SPECIAL SECTION

LEARNING ALGORITHMS AND SIGNAL PROCESSING FOR BRAIN-INSPIRED COMPUTING

- 12 FROM THE GUEST EDITORS**
Osvaldo Simeone, Bipin Rajendran, André Grüning, Evangelos S. Eleftheriou, Mike Davies, Sophie Deneve, and Guang-Bin Huang
- 16 THE IMPORTANCE OF SPACE AND TIME FOR SIGNAL PROCESSING IN NEUROMORPHIC AGENTS**
Giacomo Indiveri and Yulia Sandamirskaya
- 29 EVENT-DRIVEN SENSING FOR EFFICIENT PERCEPTION**
Shih-Chii Liu, Bodo Rueckauer, Enea Ceolini, Adrian Huber, and Tobi Delbruck
- 38 SIGNAL PROCESSING FOUNDATIONS FOR TIME-BASED SIGNAL REPRESENTATIONS**
Noyan C. Sevtiktekin, Lav R. Varshney, Pavan K. Hanumolu, and Andrew C. Singer



PG. 8



ON THE COVER

This special issue of *IEEE Signal Processing Magazine* examines the role of signal processing in neuromorphic computing and brings together key researchers to provide readers with up-to-date and survey-style articles on algorithmic, hardware, and neuroscience perspectives on the state-of-the-art aspects of this emerging field.

COVER IMAGE: ©ISTOCKPHOTO.COM/VERTIGO3D

- 51 SURROGATE GRADIENT LEARNING IN SPIKING NEURAL NETWORKS**
Emre O. Nefci, Hesham Mostafa, and Friedemann Zenke
- 64 AN INTRODUCTION TO PROBABILISTIC SPIKING NEURAL NETWORKS**
Hyeryung Jang, Osvaldo Simeone, Brian Gardner, and André Grüning
- 78 SPIKING RESERVOIR NETWORKS**
Nicholas Soares and Dhireesha Kudithipudi
- 88 NEUROSCIENCE-INSPIRED ONLINE UNSUPERVISED LEARNING ALGORITHMS**
Cengiz Pehlevan and Dmitri B. Chklovskii

97 LOW-POWER NEUROMORPHIC HARDWARE FOR SIGNAL PROCESSING APPLICATIONS

Bipin Rajendran, Abu Sebastian, Michael Schmuker, Narayan Srinivasa, and Evangelos Eleftheriou

FEATURE

111 SPEECH PROCESSING FOR DIGITAL HOME ASSISTANTS

Reinhold Haeb-Umbach, Shinji Watanabe, Tomohiro Nakatani, Michiel Bacchiani, Björn Hoffmeister, Michael L. Seltzer, Heiga Zen, and Mehrez Souden

COLUMNS

8 Special Reports

Medicine on the Move
John Edwards

125 Applications Corner

A Signal Processing Perspective of Monitoring Active Volcanoes
Muhammad Salman Khan, Millaray Curilem, Fernando Huenupán, Muhammad Farhan Khan, and Néstor Becerra Yoma



PG. 168

IEEE SIGNAL PROCESSING MAGAZINE (ISSN 1053-5888) (ISPREG) is published bimonthly by the Institute of Electrical and Electronics Engineers, Inc., 3 Park Avenue, 17th Floor, New York, NY 10016-5997 USA (+1 212 419 7900). Responsibility for the contents rests upon the authors and not the IEEE, the Society, or its members. Annual member subscriptions included in Society fee. Nonmember subscriptions available upon request. **Individual copies:** IEEE Members US\$20.00 (first copy only), nonmembers US\$241.00 per copy. Copyright and Reprint Permissions: Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limits of U.S. Copyright Law for private use of patrons: 1) those post-1977 articles that carry a code at the bottom of the first page, provided the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923 USA; 2) pre-1978 articles without fee. Instructors are permitted to photocopy isolated articles for noncommercial classroom use without fee. **For all other copying, reprint, or republication permission,** write to IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854 USA. Copyright © 2019 by the Institute of Electrical and Electronics Engineers, Inc. All rights reserved. Periodicals postage paid at New York, NY, and at additional mailing offices. **Postmaster:** Send address changes to IEEE Signal Processing Magazine, IEEE, 445 Hoes Lane, Piscataway, NJ 08854 USA. Canadian GST #125634188 **Printed in the U.S.A.**

Digital Object Identifier 10.1109/MSP.2019.2936748

133 Lecture Notes

Understanding the Basis of Graph Signal Processing via an Intuitive Example-Driven Approach
Ljubiša Stanković, Danilo P. Mandić, Miloš Daković, Ilya Kisil, Ervin Sejdić, and Anthony G. Constantinides

Elucidating the Auxiliary Particle Filter via Multiple Importance Sampling
Victor Elvira, Luca Martino, Mónica F. Bugallo, and Petar M. Djurić

The Superposition Principle of Linear Time-Invariant Systems
Ming Zhang and Anxue Zhang

157 Tips & Tricks

A Fast, Accurate, and Separable Method for Fitting a Gaussian Function
Ibrahim Al-Nahhal, Octavia A. Dobre, Ertugrul Basar, Cecilia Moloney, and Salama Ikki

168 In the Spotlight

The Industry Digital Signal Processing Technology Standing Committee
Mike Polley and Fa-Long Luo

DEPARTMENTS

3 From the Editor

Organizing a Special Issue of *IEEE SPM*
Robert W. Heath, Jr.

5 President's Message

"Guilty of the Good We Did Not Do"
Ali H. Sayed

166 Dates Ahead



©ISTOCKPHOTO.COM/ELIAN88

The IEEE International Conference on Image Processing will be held 25–28 October 2020 in Abu Dhabi, United Arab Emirates.

EDITOR-IN-CHIEF

Robert W. Heath, Jr.—The University of Texas at Austin, U.S.A.

AREA EDITORS

Feature Articles

Matthew McKay—Hong Kong University of Science and Technology, Hong Kong SAR of China

Special Issues

Namrata Vaswani—Iowa State University, U.S.A.

Columns and Forum

Rodrigo Capobianco Guido—São Paulo State University (UNESP), Brazil

Roberto Togneri—The University of Western Australia

e-Newsletter

Ervin Sejdić—University of Pittsburgh, U.S.A.

Social Media and Outreach

Tiago Henrique Falk—INRS, Canada

Special Initiatives

Andres Kwasinski—Rochester Institute of Technology, U.S.A.

Nuria Gonzalez Prelcic—Universidade de Vigo, Spain

EDITORIAL BOARD

Daniel Bliss—Arizona State University, U.S.A.
 Danijela Cabric—University of California, U.S.A.
 Volkan Cevher—École polytechnique fédérale de Lausanne, Switzerland

Mrityunjoy Chakraborty—Indian Institute of Technology, Kharagpur, India

George Christos—Qualcomm, Inc., U.S.A.

Elza Erkip—New York University, U.S.A.

Alfonso Farina—Leonardo S.p.A., Italy

B. Vikram Gowreesunker—Texas Instruments Inc., U.S.A.

Joseph Guerri—Information Systems Laboratories, Inc., U.S.A.

Nageen Himayat—Intel, U.S.A.

Clem Karl—Boston University, U.S.A.

C.-C. Jay Kuo—University of Southern California, U.S.A.

Erik Larsson—Linköping University, Sweden

David Love—Purdue University, U.S.A.

Maria G. Martini—Kingston University, U.K.

Helen Meng—The Chinese University of Hong Kong, China

Meinard Mueller—Friedrich-Alexander Universität Erlangen-Nürnberg, Germany

Phillip A. Regalia—U.S. National Science Foundation

Alejandro Ribeiro—University of Pennsylvania, U.S.A.

Douglas O'Shaughnessy—INRS Université de Recherche, Canada

Osvaldo Simeone—Kings College London, U.K.

Milica Stojanovic—Northeastern University, U.S.A.

Ananthram Swami, Army Research Labs, U.S.A.

Jong Chul Ye—KAIST, South Korea

Qing Zhao—Cornell University, U.S.A.

Josiane Zerubia—INRIA Sophia-Antipolis Méditerranée, France

ASSOCIATE EDITORS—COLUMNS AND FORUM

Ivan Bajic—Simon Fraser University, Canada
 Balázs Bank—Budapest University of Technology and Economics, Hungary

Panayiotis (Panos) Georgiou—University of Southern California, U.S.A.

Hana Godrich—Rutgers University, U.S.A.

Yuan-Hao Huang—National Tsing Hua University, Taiwan

Euee Seon Jang—Hanyang University, South Korea

Vishal M. Patel—Rutgers University, U.S.A.

Christian Ritz—University of Wollongong, Australia

Changshui Zhang—Tsinghua University, China

H. Vicky Zhao—Tsinghua University, China

ASSOCIATE EDITORS—e-NEWSLETTER

Nesreen Ahmed—Intel, U.S.A.

Behnaz Ghorani—Florida Atlantic University, U.S.A.

Anubha Gupta—IIIT Delhi, India

Yang Li—Harbin Institute of Technology, China

Alessio Medda—Georgia Tech Research Institute, U.S.A.

Irena Orovic—University of Montenegro, Podgorica

Sarah Ostadabbas—Northeastern University, U.S.A.

Ronen Talmon—Technion, Israel

ASSOCIATE EDITOR—SOCIAL MEDIA/OUTREACH

Guijin Wang—Tsinghua University, China

IEEE SIGNAL PROCESSING SOCIETY

Ali H. Sayed—President

Ahmed Tewfik—President-Elect

Fernando Pereira—Vice President, Conferences

Nikos D. Sidiropoulos—Vice President, Membership

Sergio Theodoridis—Vice President, Publications

Tülay Adalı—Vice President, Technical Directions

IEEE SIGNAL PROCESSING SOCIETY STAFF

William Colacchio—Senior Manager, Publications

and Education Strategy and Services

Rebecca Wollman—Publications Administrator

IEEE PERIODICALS MAGAZINES DEPARTMENT

Jessica Welsh, *Managing Editor*

Geraldine Krohn-Taylor,
Senior Managing Editor

Janet Dudar, *Senior Art Director*

Gail A. Schnitzer, *Associate Art Director*

Theresa L. Smith, *Production Coordinator*

Mark David, *Director, Business Development - Media & Advertising*

Felicia Spagnoli, *Advertising Production Manager*

Peter M. Tuohy, *Production Director*

Kevin Lisankie, *Editorial Services Director*

Dawn M. Melley, *Staff Director, Publishing Operations*

Digital Object Identifier 10.1109/MSP.2019.2936749



IEEE prohibits discrimination, harassment, and bullying.
 For more information, visit
<http://www.ieee.org/web/aboutus/whatis/policies/p9-26.html>



Organizing a Special Issue of *IEEE SPM*

In my September editorial [1], I outlined the important components of good feature articles in response to feedback from our recent IEEE Periodicals Review and Advisory Committee (PRAC) meeting. In this issue's editorial, I discuss the process of organizing a special issue (SI) for *IEEE Signal Processing Magazine (SPM)*. The purpose is twofold. First, I want to encourage readers to become involved in organizing an SI. Second, I want to provide more information about the decision-making process so that authors understand how they can have an article accepted.

I begin this description with some background information about SIs. An SI is a collection of articles that focuses on a specific topical area and is generated in response to an SI call for papers [2]. SI articles, like feature articles, are tutorial in nature. As is the case with feature articles [1], SI articles should not include any new ideas or fundamental results that could appear in, e.g., *IEEE Transactions on Signal Processing*. Any article that features new results is not within the scope of an SI article and will be rejected.

The composition of an SI article is similar to that of a feature article, with two important differences. First, SI articles are more focused than feature articles; this is because there will be a collection of approximately 10 articles that address different aspects of the same research area. As a result, less background and overview

of the general topic reduces the amount of overlap among articles. Second, SI articles are shorter than feature articles. When submitted, they may be up to 20 single-column double-spaced pages, including up to 10 figures and 30 references. Feature articles are allowed twice as many pages, more figures, and more references. These two aspects mean that a rejected feature article is not a good fit for resubmission to an SI (without significant reduction), nor is a rejected SI article a good fit for a regular issue (without further expansion).

Assembling an SI proposal

Before officially proposing a specific topic, I encourage prospective guest editors to discuss the appropriateness of the topic with SI Area Editor Namrata Vaswani namrata@iastate.edu, or me, *SPM*'s editor-in-chief (EIC). We can also send you a current copy of *SPM*'s *Special Issues' Guest Editors' Handbook*.

The process of initiating an SI starts with a proposal and a draft call for papers. This proposal is sent out to the Senior Editorial Board for their review. The feedback from five to 10 board members is then relayed to the proposing team, along with additional comments from the area editor and possibly the EIC. Based on that feedback, a decision is made whether to accept the SI, request a revision of the SI proposal, or an outright rejection. It is normal for one

round of revisions to occur to address board comments.

The formal proposal includes several components. The first and foremost section is the introduction, where you encourage interest in the topic and make the case for the SI being timely and relevant. A tentative list of topics should then be

An SI is a collection of articles that focuses on a specific topical area and is generated in response to an SI call for papers.

provided to help the committee understand what you have in mind in terms of the scope of the issue. The next section is a list of potential authors. We ask for this information so

that we may evaluate the likelihood that the issue will receive enough submissions. It is important to note that these authors are not "invited"; rather, they will be asked to consider a submission. The next section is a summary of the relationship of this SI to other SIs (possibly in other publications). We then ask for biographies of the guest editors. Here, you want to show that your team is qualified, has a track record in various associate editorial roles, and has broad representation. Most issues have three to five guest editors. We solicit this information to ensure that there are no competing SIs in other venues that may reduce the number of potential submissions. Finally, a draft call for papers should be given; a template can be provided by the EIC.

Characteristics of good SI topics

Unless the SI has a historical theme, the topic should be timely and relevant for the signal processing community. Such

topics may be the subject of special sessions at conferences or not. In short, they should be topics of great interest to both authors and readers in one year's time (the typical time between submission of a white paper and publication in the magazine).

It is important to refine the scope of the SI so that it is neither too narrow nor too broad. An ideal SI will receive between 25 and 40 white papers, with roughly 15 accepted for full submission, and, approximately 12 accepted eventually. Occasionally, a double SI will be permitted if there are enough high-quality white papers submitted, but this is rare.

The review of white papers is handled by the guest editors of the SI and a liaison from the Senior Editorial Board (papers by the guest editors are handled by the liaison, the area editor for SIs, or the EIC). This review is holistic: Accepted white papers must be of high quality with minimal overlap of other accepted papers, which ensures that the SI covers a diverse set of topics. In some cases, when the white paper topics are similar, the authors will be invited to consider combining their papers. Although it is not a requirement, it can be a good opportunity for authors to broaden their collaborations.

From an author's perspective

An author's first step is preparing his or her white paper. The following characteristics make for an insightful white paper:

- **Relevance:** As with feature articles, some SI submissions are out of scope.
- **Topical focus:** In particular, the paper should address a subset of material relevant for the SI, not the entire issue. An exception to this could be made if the guest editors want to prepare an introductory article for their issue.
- **Diverse author team:** For example, a paper with authors from multiple research groups will have additional merit for broad representation.

- **Author expertise:** The editorial team has more confidence in an SI submission when the authors have previously published some related work.

- **A well-written paper:** Good white papers are polished, proofread, and complete.

Even papers that have all of these features may not be invited for full submission if there are multiple papers on competing topics or magazine space constraints.

If your white paper is accepted, then you need to diligently prepare your full paper. The SI timeline is driven by the eventual publication in a specific month and year for the magazine. Guest editors have little flexibility in adjusting dates of publication to accommodate potential delays in submission. As a result, authors should be prepared to stick to the published timelines.

Upon full submission, SI articles are assigned to an associate editor for

further review. The procedure that follows is the same as for feature articles; see [1]. The associate editor is usually selected from one of the guest editors, except

in cases of conflict. At this point, the article begins the formal review process. External reviewers are solicited by experts to provide feedback on the article. The associate editor then decides on the paper based on his or her own reading and the opinions from reviewers. Manuscripts may be rejected at this stage or given an opportunity for a revision. If you are encouraged to revise your manuscript, it is critical that you address the comments from the reviewers in the paper and provide a detailed reply to reviewers as a separate PDF file. Do your best to be thoughtful in your edits and your reply, as simply arguing with the reviewers seldom leads to a positive outcome. Typically, there is only one round of major reviews.

General suggestions about preparing a good, full-length article can be found in [1]. The main considerations are the page length, number of figures, and corresponding restricted scope. As with

feature articles, consider using equations where appropriate. Additionally, put considerable thought into meaningful numerical experiments and eye-catching graphics. Common criticisms of full papers include the following:

- **Too much textbook or introductory material:** Remember that the SI collects more focused contributions and has limited length. You may find it useful to obtain a list of white papers invited for full paper submission to see which topics require more background.
- **Not polished, proofread, or missing good paragraph and section structure:** Poor writing is distracting in a tutorial paper. Be sure to write carefully, proofread, and revise your manuscript as a team prior to submission. If time permits, obtain feedback from other colleagues or students.
- **The reference list is not well chosen:** The selection of references can be challenging when limited to only 30 papers. Good papers will find a balance between citing the authors' own work (please use self-constraint) and work from other groups.
- **The paper is overly technical.** This comment occurs, for example, when there are many pages of notation and definitions which serve little purpose in conveying the key points. While we encourage mathematics in the articles, what is included must be carefully curated.

I hope that this editorial will be useful to potential SI organizers and authors. See [3] for current SI deadlines.

References

- [1] R. W. Heath, Jr., "Making a good feature article submission," *IEEE Signal Process. Mag.*, vol. 36, no. 5, pp. 3–4, 2019.
- [2] IEEE Signal Processing Society, "SPM." Accessed on: Sept. 30, 2019. [Online]. Available: <https://signalprocessingsociety.org/tags/spm>
- [3] IEEE Signal Processing Society, "Publications special issue deadlines." Accessed on: Sept. 30, 2019. [Online]. Available: <https://signalprocessingsociety.org/publications-resources/special-issue-deadlines>



SP



“Guilty of the Good We Did Not Do”

Many of us marveled in awe in March 2018 at the sight of the Ghana teacher who, using colored chalk, drew on his blackboard a snapshot of how an open window of the Microsoft Word software would look like on the screen of a computer [1]; see Figure 1. His school did not have any computers and his young students needed to prepare to pass the Ghanaian national exam, which requires familiarity with computing and information technology. I was not only moved by the action of this amazingly dedicated and patient teacher drawing his creation on the board, but also by the sight of his attentive students sitting on their desks and copying the information from the blackboard onto their notebooks. Just pause for a while and consider how revealing these images are about human nature and its restless drive for knowledge.

While we take for granted our desk computers, software packages, laptops, and smartphones, without even giving a thought as to how precious these tools are, there are eager students, with great potential, in disadvantaged regions yearning to imagine what the screen of a computer would look like! It appears to them as if computers are characters from a fictional novel. That is a shameful gap; one that continues to exist today at the dawn of the 21st century.

This past March 2019, we also witnessed a Kenyan science teacher being awarded a US\$1 million prize for being the “world’s best teacher.” This teacher spends 80% of his salary to help the

poorest students of his school buy their books and uniforms [2]. To the amazement of many, these types of stories do not only occur in villages in Africa or Asia, but also occur in wealthier countries, including in the United States of America, where Good Samaritans frequently assist their needy students. There is untapped potential in our world, with great challenges to uncover. I will focus in this editorial on Africa, knowing very well that the same challenges and potential exist elsewhere on the globe.

According to [3], Africa generates less than 1% of the world’s research while accounting for about 16% of the world population. The continent has 79 scientists per million inhabitants, compared to 4,500 for the United States [3]. These are alarming statistics if we consider that Africa needs to develop its science base to meet immense challenges that are likely to affect the continent even more seriously than other locations, especially

in relation to climate change, agriculture, health, and population growth. The lower research output in Africa is not for lack of institutions of higher education. There exist literally hundreds of universities, with those from South Africa and Egypt often appearing among the highest ranked in the continent. Nevertheless, according to a study by the Center for World University Rankings, there are only 10 universities from Africa appearing in the list of the top 1,000 universities worldwide [4]. In the opinion of many experts, the main challenge facing African universities is the lack of sufficient resources or adequate infrastructure to promote a vibrant research environment, besides low salary scales. A relatively recent study [5] suggests that over 80% of academics and students in sub-Saharan Africa have held unpaid research positions.

One should also not underestimate the consequential and damaging repercussions



FIGURE 1. Photos of Ghanaian teacher, Owura Kwadwo Hottish, drawing Microsoft Word on a blackboard have gone viral. (Source: Owura Kwadwo Hottish; used with permission.)

from a history of colonialism and slavery that continue to reverberate to this day. Many African countries have only gained independence as recently as the mid 1950s. This dynamic was acknowledged by former U.S. President Barack Obama in his speech to the people of Africa in Addis Ababa, Ethiopia, in July 2015, when he declared [6]: “So, too, here in Africa. This is the cradle of humanity, and ancient African kingdoms were home to great libraries and universities. But the evil of slavery took root not only abroad, but here on the continent. Colonialism skewed Africa’s economy and robbed people of their capacity to shape their own destiny. Eventually, liberation movements grew. And 50 years ago, in a great burst of self-determination, Africans rejoiced as foreign flags came down and your national flags went up.”

We are talking about a proud continent, rich in history and also in potential. According to the International Monetary Fund (IMF), some of the economies with the fastest rates of growth in the world are actually located in Africa, including Ethiopia, Rwanda, Ghana, Kenya, Uganda, and others [7].

The difficulties that African universities face do not mean that impactful research is not happening in Africa. There are many laudable and world-recognized efforts in various domains, especially in relation to fields of particular significance to Africa including research on HIV/AIDS, climate change, agriculture, and anthropology. For example, Glenda Gray, a South African physician known for her work on HIV research and President of the South African Medical Research Council, was named in the 2017 TIME list of 100 most influential people in the world [8]. The husband and wife professors Salim and Quarraisha Abdool-Karim are award-winning epidemiologists from South Africa, widely recognized for their work on infectious diseases. The husband is a member of the U.S. National Academy of Medicine, while the wife is a recipient of the 2016 L’Oreal-UNESCO Award for Women in Science. In 2017, she was also named by the BBC as one of the seven trailblazing women in science; this list starts with Marie Curie [9]!

And if we go back a few decades earlier, we can mention the transformative contribution by the South African surgeon Christiaan Barnard (1922–2001) who performed the first successful heart transplant in 1967. Interestingly, this is how Barnard referred to his feat, “On Saturday I was a surgeon in South Africa, very little known, and on Monday I was world renowned.” How many lives on a global scale have the works of these various individuals saved?

There is no question that talent exists in abundance in Africa, especially younger talent with the continent’s population accounting for the youngest median age in the world at 19.7 years according to *Wikipedia*. But talent this young needs to be nurtured. Given an opportunity, they can grow, shine, and compete at the highest levels. Examples are many from within the continent and also from outside.

Former U.S. President Obama’s father was a Kenyan who traveled to the United States to receive an undergraduate degree in economics from the University of Hawaii and an M.A. degree in economics from Harvard University in the 1960s. It took only one generation for the son to rise to the highest office in the United States and leave a mark on the world stage. Another prominent public figure of African descent is the former U.N. Chief Kofi Anan (1938–2018) from Ghana, who studied in the United States and Switzerland in the early 1960s. He went on to share along with the United Nations the 2001 Nobel Peace Prize for their work “for a better organized and more peaceful world.” On the science side, the South-African American physicist Allan Cormack (1924–1998) shared the 1979 Nobel Prize in Medicine for his work on X-ray CAT scans and was later awarded the U.S. National Medal of Science in 1990. He taught at Tufts University. Another example is the 1999 Chemistry Nobel Laureate Ahmed Zewail (1946–2016) who emigrated from Egypt to the United States and taught at Caltech. He is known as the “father of femtochemistry.” This list of individuals is not meant to be exhaustive in any way. They are only meant to serve as examples, while

acknowledging that many other prominent scientists and leaders exist across other countries in the African continent. Not to mention the long list of impactful scientists and leaders of African-American heritage in the United States alone.

I am a believer in the power of diversity in science. In one of my earlier editorials on the “blindness of science,” I spoke about the value of diversity and how it enriches the academic discourse [10]. I am also a believer in the power of education and how it can serve as a bridge to empower people, improve their conditions, and bring communities closer together. Everyone wins from an environment where education prevails, and where science works hand in hand to solve challenges that affect people regardless of their race, ethnicity, origin, gender, or religion. We are all subject to the consequences of climate change, independent of where we live. The same life-threatening diseases can mortally wound us all; none of us is immune by their geography or history.

For these reasons, educational and research institutions in the Western world should do more to reach out to the African continent. We cannot progress through the 21st-century thinking of conquering Mars and the stars while students in remote villages are only imagining what computers look like! How many powerful and promising minds don’t attain their full potential due to lack of opportunity? Today, when we face a computationally challenging problem, we throw more computing cores at it and launch our powerful supercomputers into full gear. But brute force computing alone cannot solve all problems. Many problems in science require an injection of a higher dose of creativity, and not of computations. The human mind is the nuclear center of creativity in this universe, something that machines, no matter how powerful or how “intelligent” they are claimed to be, will never match. We can assign more creativity to our problems by enlarging the pool of thinkers around them.

There are many ways by which western institutions of higher education can reach out to African educational institutions, including allocation of research funding for joint projects, fellowship

funding for student training, mutual and regular visits by researchers and students, offerings of educational courses and training, organization of scientific meetings, and joint publications. Many universities and foundations from the United States and Europe are already moving in this direction. The Bill and Melinda Gates Foundation invests in solutions that advance health conditions in Africa. Stanford offers Africa MBA fellowships. MIT offers the MIT-Africa program for engagement with African countries. Carnegie Mellon University offers the CMU Africa program, while EPFL in Europe offers a range of MOOC courses for African countries. The power of the online medium can be used to great advantage to facilitate interactions.

The Institute for Electrical and Electronics Engineers (IEEE) has also identified Africa as a focus area and has formed an ad-hoc committee to develop an IEEE strategy for Africa. Our former IEEE Signal Processing Society (SPS) President Rabab Ward has been a vocal proponent and regular supporter for such efforts. She even started her academic career teaching in Africa! More recently, the SPS has approved an initiative to support continuing education and outreach efforts to Africa. We have funded an effort to support visits by distinguished speakers to African countries. We have also established a new position of Vice-President Education for the Society. One of the responsibilities of the position is to develop our continuing education program, as well as enrich our online library of educational material, seminars, tutorials, and short courses, and make this wealth of information available for use in outreach efforts. We have further instituted a standing policy of US\$1 annual membership fee for all students, in an effort to make our Society affordable to students everywhere including from disadvantaged regions. We have reduced the tutorial fees at our conferences for all students to a minimal level. These steps are making a difference. At our most recent ICASSP conference held in Brighton, United Kingdom, from May 12th to 17th of this year. I was informed that the number of students enrolled in the conference has increased sixfold

compared to the previous ICASSP edition held in Calgary, Canada 2018 (750 students registered in 2019 versus 128 in 2018), thus enabling ICASSP to pass the mark of 3,000 attendees for the first time in its history. Also, the number of student memberships in our Society has doubled in one single year since instituting the US\$1 policy.

As a professional organization we need to live by the maxim that “every man is guilty of all the good he did not do.” There are many opportunities for an organization with our resources and capacity to make a meaningful difference in the world and we should. As the saying goes, “if you see someone without a smile, give them one of yours.”

References

- [1] G. Mezzofiore, “New Word order: Ghanaian teacher uses blackboard to explain software,” CNN, Mar. 1, 2018. Accessed on: Aug. 21, 2019. [Online]. Available: <https://www.cnn.com/2018/03/01/africa/ghana-teacher-blackboard-intl/index.html>
- [2] J. Otte, “Kenyan science teacher Peter Tabichi wins \$1m global award,” *The Guardian*, Mar. 24, 2019. Accessed on: Aug. 21, 2019. [Online]. Available: <https://www.theguardian.com/education/2019/mar/24/kenyan-science-teacher-peter-tabichi-wins-1m-global-award>
- [3] T. Kariuki, “Africa produces just 1.1% of global scientific knowledge—but change is coming,” *The*

Guardian, Oct. 26, 2015. Accessed on Aug. 22, 2019. [Online]. Available: <https://www.theguardian.com/global-development-professionals-network/2015/oct/26/africa-produces-just-11-of-global-scientific-knowledge>

[4] Y. Kazeem, “Only ten of the world’s top 1,000 universities are in Africa, according to one list,” *Quartz*, July 14, 2016. Accessed on: Aug. 22, 2019. [Online]. Available: <https://qz.com/africa/731712/only-ten-of-the-worlds-top-1000-universities-are-in-africa-according-to-one-list/>

[5] M. Makoni, “Research is often unpaid in sub-Saharan Africa,” *Nature*, Oct. 31, 2018. Accessed on: Aug. 21, 2019. Available: <https://www.nature.com/articles/d41586-018-07244-w>

[6] The White House, “Remarks by President Obama to the People of Africa,” July 28, 2015. Accessed on: Aug. 21, 2019. [Online]. Available: <https://obama.whitehouse.archives.gov/the-press-office/2015/07/28/remarks-president-obama-people-africa>

[7] Y. Adegoke, “Africa will have some of the world’s fastest-growing economies in 2019—and a looming debt crisis,” *Quartz*, Jan. 13, 2019. Accessed on Aug. 22, 2019. [Online]. Available: <https://qz.com/africa/1522126/african-economies-to-watch-in-2019-and-looming-debt/>

[8] S. O’Connor, “Glenda Gray,” *Time*. Accessed on: Aug. 21, 2019. [Online]. Available: <https://time.com/collection/2017-time-100/4742691/glenda-gray/>

[9] BBC News Services, “100 Women: Seven trailblazing women in science,” BBC News, Nov. 6, 2017. Accessed on: Aug. 22, 2019. [Online]. Available: <http://www.bbc.com/news/science-environment-41861232>

[10] A. H. Sayed, “Science is blind,” *IEEE Signal Process. Mag.*, vol. 35, no. 5, pp. 4–6, Sept. 2018.

Signature

SP



Professor/Associate Professor/Assistant Professorship in the Department of Electrical and Electronic Engineering

The Department of Electrical and Electronic Engineering, Southern University of Science and Technology (SUSTech) now invites applications for the faculty position in the Department of Electrical and Electronic Engineering. It is seeking to appoint a number of tenured or tenure track positions in all ranks.

Candidates with research interests in all mainstream fields of electrical and electronic engineering will be considered, including but not limited to IC Design, Embedded Systems, Internet of Things, VR/AR, Signal and Information Processing, Control and Robotics, Big Data, AI, Communication/Networking, Microelectronics, and Photonics. These positions are full time posts. SUSTech adopts the tenure track system, which offers the recruited faculty members a clearly defined career path.

Candidates should have demonstrated excellence in research and a strong commitment to teaching. A doctoral degree is required at the time of appointment. Candidates for senior positions must have an established record of research, and a track-record in securing external funding as PI. As a State-level innovative city, it is home to some of China’s most successful high-tech companies, such as Huawei and Tencent. We also emphasize entrepreneurship in our department with good initial support. Candidates with entrepreneur experience is encouraged to apply as well.

To apply, please send curriculum vitae, description of research interests and statement on teaching to eehire@sustech.edu.cn. SUSTech offers internationally competitive salaries, fringe benefits including medical insurance, retirement and housing subsidy, which are among the best in China. Salary and rank will commensurate with qualifications and experience.

More information can be found at <http://talent.sustech.edu.cn/en> and <http://eee.sustech.edu.cn/en>. The search will continue until the position is filled.

For informal discussion about the above posts, please contact Chair Professor Xiao Wei SUN, Head of Department of Electrical and Electronic Engineering, by phone 86-755-88018558 or email: sunxw@sustech.edu.cn.

To learn more about working & living in China, please visit: <http://www.jobs.ac.uk/careers-advice/country-profiles/china>.

Medicine on the Move

Wearable devices supply health-care providers with the data and insights necessary to diagnose medical issues and create optimal treatment plans

This is an age of mobility. Phones, tablets, notebook computers, smart watches, and various other devices now supply people around the world with instant communication capabilities that were only dreamed of a generation ago. Mobility technologies are also transforming medicine, helping to improve the quality of care for people at all stages of life, giving both patients and health-care providers deeper, more accurate insights into various medical conditions. An emerging generation of mobile and often wearable medical devices, enabled by sensors and signal processing, allow caregivers to noninvasively probe deeply inside the human body. The technologies are designed to analyze physical processes in real time and/or over defined periods to treat current conditions, detect hidden problems, and provide customized treatment regimens.

Wearable system monitors digestion

Gastrointestinal (GI) problems are a leading cause for missing work or school as well as a significant reason for patient visits to primary care physicians. At the University of California, San Diego (UCSD), engineers and physicians have collaboratively developed a wearable, noninvasive system to monitor electrical activity in the stomach over an entire day.

The system, a high-resolution electrogastrogram (HR-EGG), is similar in approach to an electrocardiogram (ECG), which involves placing multiple electrodes over the chest to detect generated waveforms that help clinicians interpret various heart functions. The HR-EGG (Figure 1) uses multiple electrodes placed over the abdomen to pick up waveforms that can assist clinicians in interpreting stomach functions in patients' daily lives.

The HR-EGG is designed to provide a noninvasive "electrical window" into the digestive system, says project researcher Todd P. Coleman, a UCSD bioengineering professor and the corresponding author on a research paper describing the project. "When a patient wears the

system, they can utilize an app on their phone to indicate when a meal, symptom, bathroom event, or other time point of interest has occurred," he explains.

Other project participants include Armen Gharibans, the paper's first author and a UCSD bioengineering postdoctoral researcher; David Kunkel, a gastroenterologist at UCSD Health; and Benjamin Smarr, a chronobiologist at the University of California, Berkeley. The research was funded via Larry Smarr, a professor in UCSD's Computer Science and Engineering Department as well as founding director of the California Institute for Telecommunications and Information Technology.

The researchers' biggest challenge was developing algorithms that could

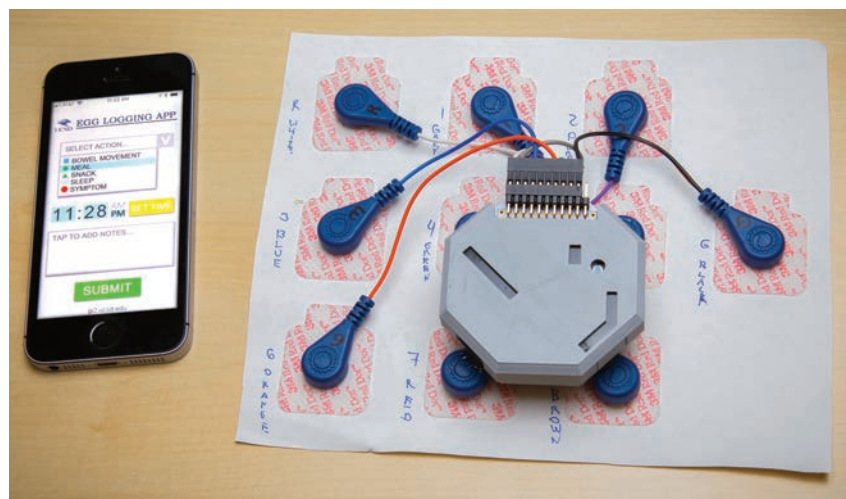


FIGURE 1. The HR-EGG, developed at UCSD, is designed to provide a noninvasive "electrical window" into a patient's digestive system. (Source: UCSD; used with permission.)

recognize and boost the stomach's electrical signals amid noise and artifacts. Since the stomach's electrical signals are 10 times weaker than the heart's, they are difficult to capture and successfully analyze. "Signal processing is crucially important, because we use arrays of electrodes over the abdomen to acquire information about the digestive system, but ultimately we need to provide descriptions about the function of the digestive system that are succinct, are easy to interpret, and have clinical significance so that they can aid in decisions about care," Coleman says.

After studying existing literature on stomach physiology, Coleman and his signal processing group noticed that the stomach generates unique wave-like electrical propagation that can go awry in patients with certain GI disorders. "This gave us the sense that perhaps we should spend more time on developing algorithms that interrelate waveforms from different electrodes, along with the positions of the waveforms, to extract spatial information involving wave propagation," he explains. "With that [knowledge], we were able to build upon recently developed wave signal processing methods developed in the neuroscience community and tailor them specifically to this setting."

Signal processing also helped the team overcome another key challenge: the fact that abdominal muscles sometimes generate very large wideband, signal-distorting artifacts. "We were able to take advantage of the fact that artifacts occur seldom in time and are very large in magnitude, whereas the signal associated with the digestive system is oscillatory of a known frequency and is very low magnitude," Coleman says. "We were able to then develop signal processing algorithms that extract interferers from the signal to drastically improve the signal-to-noise ratio of the electrical signal of the digestive system." The approach allowed the researchers to extract continuous waveforms over hours and time-synchronize them with events of interest to the digestive system.

The wearable device features electronics and a battery mounted inside a

3D-printed box. The system uses the same off-the-shelf electrodes as ECG units. The accompanying smartphone app lets patients log meals, sleep, and other routine activities. A planned app update promises to allow patients and physicians to view data collected by the device in real time.

"Our long-term vision is for this technology] to be commercialized, but it first needs to be deployed in research studies and in select clinical cases where it demonstrates a clear addition of value," Coleman says.

Wearable sensors promise improved Parkinson's disease treatment

Researchers at Florida Atlantic University (FAU) have developed wearable motion sensors that can detect and monitor medication on and off states, determining the effectiveness of the current prescription and dose in Parkinson's disease (PD) patients. PD is a chronic, progressive neurological disorder that affects some 6 million people worldwide, a number that's expected to double by 2040. The disease leads to motor impairments such as tremors, difficulty walking and balancing, sleep and speech issues, and cognitive impairments.

Patients with PD often experience motor fluctuations, defined as periods when they are *on*, responding positively to their medication, and *off*, when the patient experiences a reemergence of symptoms. Such fluctuations occur in 50% of patients within three to five years of diagnosis, climbing to 80% within a decade.

Motor fluctuations require patients to seek regular treatment adjustments, including medication dosage and frequency changes. Yet, lacking precise data on the timing and severity of symptoms, physicians are generally left to make educated guesses on the best treatment choices. The FAU researchers (Figure 2) have created an algorithm and a wearable sensor-based system that can detect on- and off-state patterns in PD patients. A pair of inertial measurement unit (IMU) sensors, placed on an upper and a lower extremity of the PD patient, collect movement signals as daily activities, such as walking and getting dressed, are performed, providing objective measurements of disease severity and helping the attending physician to develop a patient-specific treatment plan.

The battery-powered sensors transmit data wirelessly to a nearby mobile



FIGURE 2. Behnaz Ghoraani (right), an assistant professor in FAU's Computer and Electrical Engineering Department, and graduate student Murtadha Hssayeni examine the signal processing algorithm they developed to noninvasively monitor the state of Parkinson's disease patients. The algorithm generates a report that can be used by the treating physician to determine how a patient responds to medication. (Source: FAU; used with permission.)

phone that, in turn, relays the information to a server. A signal processing machine-learning algorithm transforms the data into a report on how the patient is responding to medication. The report is then sent to the treating physician for review.

The technology promises to enhance health care for PD patients by providing the treating physician with clinically relevant information that can be used to prescribe an effective treatment plan, notes Behnaz Ghoraani, an assistant professor in FAU's Computer and Electrical Engineering Department. "The medication can be adjusted individually according to the patients' response to medication," she reports. "Also, the system could reduce the burden on the patient by reducing the need for unnecessary on-site clinical visits."

The system is expected to significantly improve the success of PD care, streamline the assessment process, optimize treatment, and increase the likelihood of delivering optimal support to patients in rural areas where specialty care is less accessible. "It could also be used by [pharmaceutical researchers] creating new medications for PD patients," Ghoraani explains. "They could use the system to assess how patients respond to different medications."

Signal processing machine learning is central to the technology. "It enables the system to translate the body motion

data into clinically actionable information on how the patient responded to medication," Ghoraani says. The main challenge, she adds, is developing algorithms in such a way that they represent only the patterns of PD and not the patterns of activities of daily living (ADL).

The first signal processing step is converting the IMU data to the short-time Fourier transform (STFT) domain. "Next, we use multidimensional signal processing techniques to analyze the STFTs of the two sensors," Ghoraani notes. The technique is based on tensor decomposition, a multidimensional signal processing technique. "It enables us to filter out the PD-related signal patterns from the patterns of ADL as the subjects move in their natural environment."

The next step is extracting selected signal features to represent different PD symptoms while the medication is not working as well as motor complications that could happen while the medication is working. "We use signal processing methods such as sample entropy, signal mean, and standard deviation, 4–6-Hz signal power, to represent the signal features," Ghoraani says. Finally, the extracted features are used to train a machine-learning model to detect when the medication is working (medication on) and when it's not working (medication off).

Individualizing the models is the researchers' next focus. The current

machine-learning models are trained for a specific set of patients and then applied to new patients as they arrive. However, one subject's on state could be another subject's off state, depending on the disease stage and each individual's unique symptom set. Individualized models would allow more accurate treatment plans.

The project's long-term goal is developing a customized therapeutic management system that can fully automate and optimize therapy adjustments. "We are working on developing new algorithms that can use the sensor-based assessment data to rate the severity degree of change in medication response," Ghoraani says.

Monitoring joint-surgery patient progress

A self-powered sensor developed at Canada's University of Waterloo promises to help physicians and other caregivers remotely monitor patients recovering from joint surgery. The small, tube-like device, as yet unnamed, is designed to be fitted to braces after surgery to wirelessly send information to computers, smartphones, or smart watches to help caregivers track range of motion and other improvement indicators.

"We developed a sensor that's meant to monitor the healing process of human joints," says project researcher Hassan Askari, a recent Ph.D. graduate from the University of Waterloo. While the device is initially designed for monitoring knee motion, it can also be easily modified to sense other joints, such as the elbow, Askari notes (Figure 3). The sensor also has potential applications in a number of areas beyond health care, including monitoring the tires of autonomous vehicles to detect and respond to icy roads.

The device is based on a pair of electricity generation mechanisms. One of the units is an electromagnetic generator (EMG). As the wearer's joint bends, relative motion between a conductive coil and a permanent magnet is introduced, generating electricity. The other unit produces voltage through a triboelectric nanogenerator (TENG), a new technology designed for both sensing and energy harvesting applications. "The main idea is that the EMG unit will scavenge

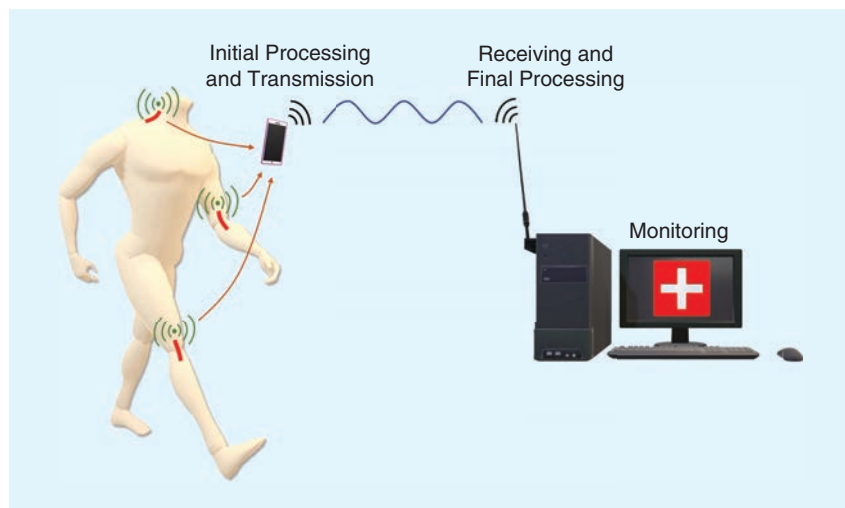


FIGURE 3. A noninvasive self-powered sensor developed at Canada's University of Waterloo links to a smartphone to help physicians and other caregivers remotely monitor patients recovering from joint surgery. (Source: University of Waterloo; used with permission.)

required power for wirelessly sending the data, and the TENG unit plays the sensing role to extract high-resolution data related to joint movement,” Askari explains. “Triboelectricity is created when specific types of material rub against each other—what we commonly think of as static electricity,” he notes. The device’s TENG is made out of polyurethane and Kapton, a polymer film. “By bending the two layers and bringing the two layers together, we make electricity flow,” Askari says.

EMGs are low-voltage, high-current sensors, while TENGs tend to be high-voltage, low-current sensors. EMGs are also able to provide high amounts of power. Therefore, the researchers opted to design a hybrid sensor in which the EMG is used for energy scavenging, while the TENG signal output is dedicated to sensing purposes. “Using this strategy, you can have a measured signal that can go as high as 10 V or even more,” Askari says.

The sensor is completely self-powered and requires no maintenance, meaning it can be used for as long as necessary. “It only depends on the patient’s condition and the specific usage directions given by the specialist,” Askari notes. He estimates that the sensor, which measures approximately 6 cm long and 1 cm wide, could be commercially manufactured for between CAN\$5 and CAN\$10.

Due to the irregular patterns of human motions as well as the nature of the sensing mechanism itself, signals transmitted by the device can be relatively noisy. “Therefore, it’s very important to consider different techniques, such as analog/digital filters, to remove the noises,” Askari says. Additionally, depending on the application, data may need to be downsampled and compressed before transmission.

Askari says the researchers worked hard to find the most appropriate noise-filtering approach. “To avoid any possible noise, taking into account different possible causes of possible noises and their frequency ranges, we have designed a combined high-/low-pass filtering for the data acquisition (DAQ) of the sensor,” he explains. “This will help to avoid noises between a possible frequency range.”

Signal processing also plays an important role in data analysis. “Using Fourier transforms can easily help us in finding some information about the joint movements, such as speed and amplitude,” Askari notes. “However, we are moving toward using intelligent methods not only to obtain that information, but also to be able to classify the status of the joint health ... based on its movement.”

Since the device’s TENG component is able to produce a relatively high voltage, no amplification is required. “In rare cases, a voltage divider might be needed to ensure that the generated high-voltage/low-current signals are within the range of the DAQ that we are using,” Askari acknowledges. Once the data has been cleaned and prepared for use, machine-learning algorithms are applied to interpret the information. “This is the part that we plan for further investigation,” he notes.

Beyond signal processing, the biggest challenge the researchers faced was

ensuring the fabricated sensor’s durability. “Our sensors are handmade at this point, and thus they are not as durable as they are expected to be,” Askari says. “But we believe that an automated product line would enhance the sensor’s hardware quality and thus its durability.”

Project collaborators include Ehsan Asadi, a University of Waterloo Ph.D. graduate; Amir Khajepour and Mir Behrad Khamesee, both engineering professors at the University of Waterloo; Zia Saadatnia, a Ph.D. candidate at the University of Toronto; and Jean Zu, a former professor at the University of Toronto and currently dean of engineering at the Stevens Institute of Technology.

Author

John Edwards (jedwards@johnedwardsmedia.com) is a technology writer based in the Phoenix, Arizona, area. Follow him on Twitter @TechJohnEdwards.

SP



Training Data for Machine Learning

Magic Data Technology is an One-stop AI Data Service Solution provider. We are committed to providing a wide range of data services in the fields of automatic speech recognition (ASR), text to speech (TTS), computer vision recognition and Natural Language Processing (NLP).

Why us:

- ✓ Efficiency: Human-in-the-loop data processing, 300,000+ professional annotators around the world
- ✓ Pioneering: Task-segmentation process and strict project management; innovative data tool software
- ✓ Quality: Our value proposition is 97%-99% quality, speed and scale (50+ languages covered)
- ✓ Professional: Multilingual & Multidomain; 100,000+ hours of data acquisition and annotation;

Two types of service:

- ✓ Over 100,000-hour self-owned copyright training data sets for building AI models quickly
- ✓ Customized data service solutions, including design, collection, annotation and processing.

Website: <http://en.imagicdatatech.com>

Linkedin: <https://www.linkedin.com/company/magicdata>

Business contact: +86 10-85527250

business@magicdatatech.com

Learning Algorithms and Signal Processing for Brain-Inspired Computing

The success of artificial neural networks (ANNs) in carrying out various specialized cognitive tasks has brought renewed efforts to apply machine learning (ML) tools for economic, commercial, and societal aims, while also raising expectations regarding the advent of an artificial “general intelligence” [1]–[3]. Recent highly publicized examples of ML breakthroughs include the ANN-based algorithm AlphaGo, which has proven capable of beating human champions at the complex strategic game of Go. The emergence of a new generation of ANN-based ML tools has built upon the unprecedented availability of computing power in data centers and cloud computing platforms. For example, the AlphaGo Zero version required training more than 64 GPU workers and 19 CPU parameter servers for weeks, with an estimated hardware cost of US\$25 million [4]. OpenAI’s video game-playing program needed training for an equivalent of 45,000 years of game play, costing millions of dollars in rent access for cloud computing services [2].

Recent studies have more generally quantified the requirements of ANN-based models in terms of energy, time, and memory consumption in both the training and inference (run-time) phases. An example is a recent work by researchers from the University of Massachusetts Amherst [5], which conclud-

ed that training a single ANN-based ML model can emit as much carbon as five cars during their lifetimes.

The massive resource requirements of ANN-based ML raise important questions regarding the accessibility of the technology to the general public and to smaller businesses. Furthermore, they pose an important impediment to deploying powerful ML algorithms on low-power mobile or embedded devices.

The importance of developing suitable methods to implement low-power artificial intelligence on mobile and embedded devices is attested by its central role in applications such as digital health, the tactile Internet, smart cities, and smart homes. In light of this, key industrial players, including Apple, Google, Huawei, and IBM, are investing in developing new chips optimized for streaming matrix arithmetic that promise to make ANN-based inference more energy efficient through complexity-reduction techniques such as quantization and pruning [6].

Neuromorphic, or brain-inspired, computing

In contrast to ANNs, the human brain is capable of performing more general and complex tasks at a miniscule fraction of the power, time, and space required by state-of-the-art supercomputers. An emerging line of work, often collectively called *neuromorphic computing*, aims at uncovering novel computational frameworks that mimic the operation of the brain, in a quest for

orders-of-magnitude improvements in terms of energy efficiency and resource requirements.

The unmatched efficiency of the human brain as an adaptive learning and inference machine may be the result of a number of unique factors. Among these, none appears to be more fundamental, and more fundamentally different from the operation of digital computer, than the way in which neurons encode information: *with* time, rather than merely *over* time [7]. Biological neurons can be thought of as complex dynamic systems with internal analog dynamics that communicate through the timing of all-or-nothing—and hence digital—spikes. This is in stark contrast to the static analog operation of neurons in an ANN. Biological neurons are connected through networks characterized by large fan-out, feedback, and recurrent signaling paths, unlike the feedforward or chain-like recurrent architectures of ANNs. As studied in theoretical neuroscience, the sparse, dynamic, and event-driven operation of biological neurons makes it possible to implement complex online adaptation and learning mechanisms via local synaptic plasticity rules and minimal energy consumption.

Based on these observations, brain-inspired neuromorphic signal processing and learning algorithms and hardware platforms have recently emerged as low-power alternatives to energy-hungry ANNs. Unlike conventional neural networks, spiking neural networks (SNNs) are trainable dynamic systems that make

use of the temporal dimension, not just as a neutral substrate for computing but also as a way to encode and process information in the form of asynchronous spike trains. In SNNs, sparse spiking and hence time-encoded signals carry out interneuron communications and intraneuron computing.

This has motivated the development of prototype neuromorphic hardware platforms that are able to process time-encoded data. These platforms include IBM's TrueNorth, SpiNNaker, developed within the "Human Brain Project"; Intel's Loihi; and proof-of-concept prototypes based on nanoscale memristive devices. These systems are typically based on hybrid digital-analog circuitry and in-memory computing, and they have already provided convincing proof-of-concept evidence of the remarkable energy savings that can be achieved with respect to conventional neural networks. Furthermore, SNNs have the unique advantage of being able to natively process spiking data as it is produced by emerging audio and video sensors inspired by biology, such as silicon cochleas or dynamic vision sensor cameras.

The role of signal processing in neuromorphic computing

Work on neuromorphic computing has been carried out by researchers in ML, computational neuroscience, and hardware design, often in parallel. While the problems under study—regression, classification, control, and learning—are central to signal processing, the signal processing community, by and large, has not been involved in the definition of this emerging field. Nevertheless, with the increasing availability of neuromorphic chips and platforms, the guest editors believe that progress in the field of neuromorphic computing calls for an interdisciplinary effort by researchers in signal processing in concert with researchers in ML, hardware design, system design, and computational neuroscience.

From a signal processing perspective, the specific features and constraints of neuromorphic computing platforms open interesting new problems concerning regression, classification, control,

and learning. In particular, SNNs consist of asynchronous distributed architectures that process sparse binary time series by means of local spike-driven computations, local or global feedback, and online learning. Ideally, they are characterized by a graceful degradation in performance as the network's number of spikes, and hence, the energy usage, increases. For example, recent work has shown that SNNs can obtain satisfactory solutions of the sparse regression problem much more quickly than conventional iterative algorithms [8]. Solutions leverage tools that are well known to signal processing researchers, such as variational inference, nonlinear systems, and stochastic gradient descent.

In this issue

The field's scope encompasses neuroscience, hardware design, and ML, which makes it difficult for a nonexpert to find a suitable entry point in the literature. This special issue brings together key researchers in this area to provide readers of *IEEE Signal Processing Magazine* with up-to-date and survey-style articles on algorithmic, hardware, and neuroscience perspectives on the state-of-the-art aspects of this emerging field.

Overview

The first special issue article, "The Importance of Space and Time for Signal Processing in Neuromorphic Agents," by Indiveri and Sandamirskaya, introduces the role of time-encoded information and parallel neuromorphic computing architectures in enabling more efficient learning agents as compared to state-of-the-art ANNs.

Sensing and time-encoded representations

Neuromorphic computing architectures take as inputs time-encoded signals that are either produced by neuromorphic sensors or converted from natural signals such as images, video, or audio. The next two articles in this special issue describe these two scenarios. In "Event-Driven Sensing for Efficient Perception" by Liu et al., the authors discuss the main properties of the data produced by neuromorphic sensors and show how

these features enable energy-efficient, low-latency, and real-time computing on neuromorphic platforms. In their article, "Signal Processing Foundations for Time-Based Signal Representations," Sevüktekin et al. discuss signal processing foundations for time-based signal representations of exogenous signals and for the reconstruction of these signals from their time-encoded versions.

Learning and signal processing applications

Neuromorphic platforms can be trained to carry out a variety of inference and control tasks. The next set of articles review training algorithms and applications. In "Surrogate Gradient Learning in Spiking Neural Networks," Neftci, Mostafa, and Zenke review training algorithms for standard deterministic models of SNNs via surrogate gradient methods, which aim at overcoming the nondifferentiability of the relevant loss functions. The next article, "An Introduction to Probabilistic Spiking Neural Networks," by Jang et al., discusses an alternative solution that is based on the use of probabilistic models by reviewing the resulting learning rules and applications. To further reduce the complexity of training, reservoir-computing techniques have been proposed, which are based on adapting only a subset of weights while others are randomly selected. Soures and Kudithipudi next present an overview of this topic in "Spiking Reservoir Networks." Finally, in "Neuroscience-Inspired Online Unsupervised Learning Algorithms," Pehlevan and Chklovskii focus on the special class of unsupervised learning algorithms, for which they provide a principled derivation of similarity-based local learning rules that are applied to problems such as linear dimensionality reduction, sparse or nonnegative feature extraction, and blind nonnegative source separation.

Hardware platforms

Standard computing systems based on the von Neumann architecture are not well suited to harness the efficiency of computing in SNNs. In "Low-Power Neuromorphic Hardware for Signal

Processing Applications,” Rajendran et al. review architectural and system-level design aspects that underlie the operation of neuromorphic computing platforms for efficient implementation of SNNs.

Guest Editors



Osvaldo Simeone (osvaldo.simeone@kcl.ac.uk) received his M.Sc. degree (with honors) and Ph.D. degree in information engineering from the Politecnico di Milano, Italy, in 2001 and 2005, respectively. He is a professor of information engineering with the Centre for Telecommunications Research, Department of Engineering, King’s College London. From 2006 to 2017, he was a faculty member at the New Jersey Institute of Technology. He is a corecipient of the 2019 IEEE Communication Society Best Tutorial Paper Award, the 2018 IEEE Signal Processing Best Paper Award, the 2017 Best Paper by the *Journal of Communication and Networks*, the 2015 IEEE Communication Society Best Tutorial Paper Award, and the IEEE International Workshop on Signal Processing Advances in Wireless Communications 2007 and IEEE Conference on Wireless Rural and Emergency Communications 2007 Best Paper Awards. He was awarded a Consolidator Grant by the European Research Council in 2016. He is a Fellow of the IEEE and of the Institution of Engineering and Technology.

He is a corecipient of the 2019 IEEE Communication Society Best Tutorial Paper Award, the 2018 IEEE Signal Processing Best Paper Award, the 2017 Best Paper by the *Journal of Communication and Networks*, the 2015 IEEE Communication Society Best Tutorial Paper Award, and the IEEE International Workshop on Signal Processing Advances in Wireless Communications 2007 and IEEE Conference on Wireless Rural and Emergency Communications 2007 Best Paper Awards. He was awarded a Consolidator Grant by the European Research Council in 2016. He is a Fellow of the IEEE and of the Institution of Engineering and Technology.



Bipin Rajendran (bipin@njit.edu) received his B.Tech. degree in instrumentation from the Indian Institute of Technology Kharagpur in 2000 and his M.S. and Ph.D. degrees in electrical engineering from Stanford University, California, in 2003 and 2006, respectively. He is an associate professor of electrical and computer engineering at the New Jersey Institute of Technology. He was a master inventor and research staff member at the IBM T.J. Watson Research Center, New York, from 2006 to 2012 and a faculty member in

the Electrical Engineering Department, the Indian Institute of Technology Bombay, from 2012 to 2015. His research focuses on building scalable architectures and systems for neuromorphic computing using nanoscale devices. He has authored more than 75 papers in peer-reviewed journals and conferences and received 59 U.S. patents, four of which were awarded IBM’s high-value patent award. His research was supported by the U.S. National Science Foundation, Semiconductor Research Corporation, and companies such as Intel and IBM. He is a Senior Member of the IEEE.

the Electrical Engineering Department, the Indian Institute of Technology Bombay, from 2012 to 2015. His research focuses on building scalable architectures and systems for neuromorphic computing using nanoscale devices. He has authored more than 75 papers in peer-reviewed journals and conferences and received 59 U.S. patents, four of which were awarded IBM’s high-value patent award. His research was supported by the U.S. National Science Foundation, Semiconductor Research Corporation, and companies such as Intel and IBM. He is a Senior Member of the IEEE.



André Grüning (a.gruning@surrey.ac.uk) received his Ph.D. degree in computer science from the Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany, working with the Complex Systems Group. He is a senior lecturer in computational intelligence in the Department of Computer Science, University of Surrey, United Kingdom, as well a visiting researcher at the Institute of Physiology, University of Bern, Switzerland. He is a task leader in the H2020 E.U. flagship project “The Human Brain Project,” where he and his team work on implementing biologicals for spiking neural networks on neuromorphic hardware. His research focuses on computational and cognitive neuroscience; he has more than 10 years of experience in designing and analyzing bioinspired learning algorithms for rate-based and spiking neural networks. He is a member of the European Institute for Theoretical Neuroscience, Paris, where he recently organized the “From Neuroscience to Machine Learning” international workshop.

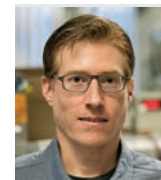
He is a senior lecturer in computational intelligence in the Department of Computer Science, University of Surrey, United Kingdom, as well a visiting researcher at the Institute of Physiology, University of Bern, Switzerland. He is a task leader in the H2020 E.U. flagship project “The Human Brain Project,” where he and his team work on implementing biologicals for spiking neural networks on neuromorphic hardware. His research focuses on computational and cognitive neuroscience; he has more than 10 years of experience in designing and analyzing bioinspired learning algorithms for rate-based and spiking neural networks. He is a member of the European Institute for Theoretical Neuroscience, Paris, where he recently organized the “From Neuroscience to Machine Learning” international workshop.



Evangelos S. Eleftheriou (ele@zurich.ibm.com) received his diploma of engineering from the University of Patras, Greece, in 1979 and his M.Eng and Ph.D. degrees

in electrical engineering from Carleton University, Ottawa, Canada, in 1985. He is currently responsible for the neuromorphic computing activities of IBM Research–Zürich. He has authored or coauthored more than 200 publications. He was a corecipient of the 2003 IEEE Communications Society Leonard G. Abraham Paper Award. He was also a corecipient of the Eduard Rhein Foundation 2005 Technology Award. In 2005, he was appointed an IBM fellow and was also inducted into the IBM Academy of Technology. In 2009, he was a corecipient of the IEEE Control Systems Society Control Systems Technology Award and the IEEE Transactions on Control Systems Technology Outstanding Paper Award. In 2016, he received an honoris causa professorship from the University of Patras, Greece. In 2018, he was inducted into the U.S. National Academy of Engineering as foreign member. He is a Fellow of the IEEE.

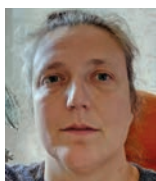
in electrical engineering from Carleton University, Ottawa, Canada, in 1985. He is currently responsible for the neuromorphic computing activities of IBM Research–Zürich. He has authored or coauthored more than 200 publications. He was a corecipient of the 2003 IEEE Communications Society Leonard G. Abraham Paper Award. He was also a corecipient of the Eduard Rhein Foundation 2005 Technology Award. In 2005, he was appointed an IBM fellow and was also inducted into the IBM Academy of Technology. In 2009, he was a corecipient of the IEEE Control Systems Society Control Systems Technology Award and the IEEE Transactions on Control Systems Technology Outstanding Paper Award. In 2016, he received an honoris causa professorship from the University of Patras, Greece. In 2018, he was inducted into the U.S. National Academy of Engineering as foreign member. He is a Fellow of the IEEE.



Mike Davies (mike.davies@intel.com) received his B.S. and M.S. degrees from the California Institute of Technology, Pasadena,

in computer engineering, mathematics, and electrical engineering. Since joining Intel Labs in 2014, he has researched neuromorphic prototype architectures and is responsible for Intel’s recently announced Loihi research chip. He leads Intel’s Neuromorphic Computing Lab. His group’s research interests span asynchronous circuits, fine-grain parallel computing architectures, neuromorphic algorithms, and software frameworks for event-driven distributed systems. He began his career in 2000 as a founding employee of Fulcrum Microsystems. As Fulcrum’s director of silicon engineering, he pioneered high-performance asynchronous design methodologies as applied to several generations of industry-leading Ethernet switch products.

Sophie Deneve (sophie.deneve@ens.fr) received two B.S. degrees, in



mathematics and biology; an M.S. degree in cognitive science; and a Ph.D. degree from the University of Rochester. She is a director of research and group leader in the Department of Cognitive Science, Ecole Normale Supérieure (ENS), Paris. Her research interests focus on how biological neural circuits learn and solve probabilistic problems, as well as probabilistic approaches to human perception and psychiatry. Her recent results include networks learning to compute as efficiently and robustly as possible with spikes, entirely derived from an objective function combining errors in estimation (both in reproducing trained example and generalizing to new ones) and cost in number of spikes. She was awarded a Dorothy Hodgkin Fellowship from Royal Society London in 2004, a Marie Curie Team of Excellence fellowship in 2006, a MacDonnell Foundation Award in 2012, and a Brain and Human Cognition grant and a European Council Consolidator grant (from 2012 to 2017).



Guang-Bin Huang (egbhuang@ntu.edu.sg) received his B.Sc. and M.Eng. degrees from Northeastern University, Shenyang, China, and his Ph.D. degree from Nanyang Technological University, Singapore. He is a full professor in the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. He is a principal investigator with the BMW-NTU Joint Future Mobility Lab on Human-Machine Interface and Assisted Driving; a principal investigator of data and video analytics with the Delta-NTU Joint Lab; a principal investigator of scene understanding with ST Engineering-NTU Corporate Lab; and a principal investigator for marine data analysis and prediction for autonomous vessels with the Rolls Royce-NTU Corporate Lab. He serves as an associate editor of *Neurocomputing*, *Cognitive Computation*, *Neural Networks*, and *IEEE Transactions on Cybernetics*. He has led or implemented several key industrial projects. He is a member of

Elsevier's Research Data Management Advisory Board and of China's International Robotic Expert Committee.

References

- [1] J. Brockman, *Possible Minds*. East Rutherford, NJ: Penguin Press, 2019.
- [2] C. Metz, "With \$1 billion from Microsoft, an A.I. lab wants to mimic the brain," *New York Times*, 2019. [Online]. Available: <https://www.nytimes.com/2019/07/22/technology/open-ai-microsoft.html>
- [3] J. Lovelock, *Novacene: The Coming Age of Hyperintelligence*. East Rutherford, NJ: Penguin Press, 2019.
- [4] AlphaGo Zero. [Online]. Available: <https://en.wikipedia.org/wiki/AlphaGo>
- [5] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," in *Proc. 57th Annu. Meeting the Association Computational Linguistics*, 2019. [Online]. Available: <http://arxiv.org/abs/1906.02243>
- [6] S. S. Sarwar, G. Srinivasan, B. Han, P. Wijesinghe, A. Jaiswal, P. Panda, A. Raghunathan, and K. Roy, "Energy efficient neural computing: A study of cross-layer approximations," *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 4, pp. 796–809, Dec. 2018. doi: 10.1109/JETCAS.2018.2835809.
- [7] F. Rieke, D. Warland, R. de Ruyter van Steveninck, and W. Bialek, *Spikes: Exploring the Neural Code*. (Computational Neurosciences series). Cambridge, MA: MIT Press, 1997.
- [8] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, et al., "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Microw.*, vol. 38, no. 1, pp. 82–99, 2018. doi: 10.1109/MM.2018.112130359.

SP

IEEE Signal Processing MAGAZINE



Twitter

LinkedIn

Interested in learning about upcoming SPM issues or open calls for papers?

Want to know what the SPM community is up to?

Follow us on twitter (@IEEEspmm) and/or join our LinkedIn group

(www.linkedin.com/groups/8277416/) to stay in the know and share your ideas!

The Importance of Space and Time for Signal Processing in Neuromorphic Agents

The challenge of developing low-power, autonomous agents that interact with the environment



©ISTOCKPHOTO.COM/JUST_SUPER

Artificial neural networks (ANNs) and computational neuroscience models have made tremendous progress, enabling us to achieve impressive results in artificial intelligence applications, such as image recognition, natural language processing, and autonomous driving. Despite this, biological neural systems consume orders of magnitude less energy than today's ANNs and are much more flexible and robust. This adaptivity and efficiency gap is partially explained by the computing substrate of biological neural processing systems that is fundamentally different from the way today's computers are built. Biological systems use in-memory computing elements operating in a massively parallel way rather than time-multiplexed computing units that are reused in a sequential fashion. Moreover, the activity of biological neurons follows continuous-time dynamics in real, physical time instead of operating on discrete temporal cycles abstracted away from real time.

Here, we present neuromorphic processing devices that emulate the biological style of processing by using parallel instances of mixed-signal analog/digital circuits that operate in real time. We argue that this approach brings significant advantages in efficiency of computation and show examples of embodied neuromorphic agents that use such devices to interact with the environment and exhibit autonomous learning.

Introduction

Tremendous progress in machine learning and computational neuroscience is leading to the development of neural processing algorithms that may have a far-reaching impact on our daily lives [1]. For example, recently developed deep and convolutional neural network algorithms can be trained to perform remarkably well in pattern-recognition tasks, in some cases even outperforming humans. Typically, these algorithms run on conventional computing systems based on the von Neumann architecture and, consequently, are commonly implemented using large and power-hungry platforms, sometimes distributed across multiple machines in server farms. The power required to run these algorithms and achieve impressive results is orders of magnitude larger than the power

used by biological nervous systems that once served as inspiration for the ANNs. This high power consumption is not sustainable for the future needs of ubiquitous computing at scale.

The reasons for the large gap in energy costs between artificial and natural neural processing systems are still being investigated; however, it is clear that one fundamental difference lies in the way the elementary computing processes are organized: Conventional computers use Boolean logic, bit-precise digital representations, and time-multiplexed and clocked operations. Nervous systems, on the other hand, carry out robust and reliable computation using analog components that are inherently noisy and operate in continuous time and activation domains. These components typically communicate among each other with all-or-none discrete events (spikes), thus, using a combination of analog computation and digital communication. Moreover, they form distributed, event-driven, and massively parallel systems, and they feature a considerable amount of adaptation, self-organization, and learning, with dynamics that operate on a multitude of timescales.

One promising approach for bridging the efficiency gap between the biological and artificial neural systems is to develop a new generation of ultralow-power and massively parallel computing technologies that are optimally suited to implement neural network architectures that use the same principles of computation used by the brain. This “neuromorphic engineering” approach [2] was originally proposed in the early 1990s [3] but now takes advantage of the progress made in very large-scale integration (VLSI) technologies to scale up the number of neurons and synapses per device, as well as the development in computational neuroscience to implement more complex, spike-based signal processing and learning architectures. Neuromorphic circuits are typically designed using analog, digital, or mixed-mode analog/digital CMOS transistors, and are well suited for exploiting the features of emerging memory technologies and new memristive materials [4]–[6]. Similar to the biological systems they model, neuromorphic systems process information using energy-efficient, asynchronous, event-driven methods [7]; they are often adaptive and fault-tolerant, and they can be flexibly configured to display complex behaviors by combining multiple instances of simpler elements.

Remarkable neuromorphic computing platforms have been developed in the past for modeling cortical circuits, solving pattern recognition problems, and implementing machine learning tasks [8]–[19], as well as for accelerating the simulation of computational neuroscience models [15], [17]. In parallel, impressive, full-custom dedicated accelerators have been proposed for implementing convolutional and deep network algorithms following the more conventional digital design flow [20], [21]. While these systems significantly reduce the power consumption in a wide variety of neural network applications compared to conventional computing approaches, they still fall short of reproducing the power, size, and adaptivity of biological neural processing systems that can produce adaptive behavior that consumes just a few watts of power. Indeed, developing low-power, compact, and autonomous electronic agents that can interact with the environment in real time is

still an open challenge. Such agents must be capable to extract relevant information from the signals they sense, adapt to the changes and uncertain conditions present in the external inputs and internal states, and learn to produce context-dependent behaviors for carrying out goal-directed procedural tasks.

We argue that to address this challenge and find a computational substrate that minimizes size and power consumption in real-time behaving systems, the computing hardware in which computation is realized needs to match the computing principles underlying autonomous adaptive behavior. In particular, to fully benefit from the emulation of biological neural processing systems, it is important to preserve two of their fundamental characteristics: the explicit representation of time and the explicit use of space, instantiating dedicated physical circuits for each neuron/synapse element and avoiding the use of time multiplexing. In the following, we first present the design principles for building neuromorphic electronic systems that make use of these representations and then show how such explicit representation of time and space matches a computing framework of dynamic neural fields that embody principles of autonomous behavior generation and learning [22], [23]. Finally, we demonstrate successful realizations of autonomous neural-dynamic architectures in neuromorphic hardware, emphasizing the role of using physical time and space representation in hardware for efficiency of the autonomous neuronal controllers.

Design principles for building biologically plausible neuromorphic processors

Dedicated neuronal circuits in hardware neural-processing systems

In an effort to minimize silicon-area consumption, digital neuromorphic processors typically use time-multiplexing techniques to share circuits that simulate neural dynamics for modeling multiple neurons [16], [18], [19]. This requires that the shared circuits continuously transfer their state variables to and from an external memory block at each update cycle (therefore, burning extra power for the data transfer). The faster the transfer and cycle rate, the larger the number of neurons that can be simulated per time unit. In addition, if these circuits need to model processes that evolve continuously over natural time, such as the leak term of a leaky integrate and fire (I&F) neuron model, it is necessary to include a clock to update the related state variables periodically and manage the passing of time (thus, adding extra overhead and power consumption).

Unlike digital simulators of neural networks, analog neuromorphic circuits use the physics of silicon to directly emulate neural and synaptic dynamics [2]. In this case, the state variables evolve naturally over time, and “time represents itself” [3], bypassing the need to have clocks and extra circuits to manage the representation of time. Furthermore, since the state variable memory is held in the synapse and neuron capacitors, there is no need to transfer data to extra memory blocks, dramatically saving energy that would otherwise be required to shift the neuron state variables back and forth from the memory. Examples of neuromorphic architectures that follow this

mixed-signal approach include the Italian Istituto Superiore di Sanità (ISS) learning attractor neural network that has plasticity and a long-term memory chip (LANN-21) [8]; ISS “final learning attractor neural network” (F-LANN) chip [9]; Georgia Tech learning-enabled neuron array integrated circuit (LENA-IC) [10]; University of California San Diego integrate-and-fire array transceiver (IFAT) architecture [11]; Stanford University, California, Neurogrid system [12]; University of Zürich recurrent online learning spiking (ROLLS) neuromorphic processor [14]; and University of Zürich dynamic neuromorphic asynchronous processor- scalable (DYNAP-SE) chip [13].

In these devices, the analog synapse and neuron circuits have no active components [2]. The circuits are driven directly by the input “streaming” data. Their synapses receive input spikes, and their neurons produce output spikes at the rate of the incoming data. So, if they are not processing data, there is no energy dissipated per synaptic operation (SOP) and no dynamic power consumption. Therefore, this approach is particularly attractive in the case of applications in which the signals have sparse activity in space and time. Under these conditions, most neurons would be silent at any one time, thus, bringing the system power consumption to the minimum.

A quantitative comparison of the specifications of these devices is presented in Table 1. The spike-based learning algorithms that some of these devices implement are based on the basic version of spike-timing-dependent plasticity (STDP) [24] or the more elaborate spike-timing and rate-dependent plasticity (STRDP) [25].

While this approach is very power efficient, it is necessary to instantiate and place one distinct physical neuromorphic circuit per emulated synapse or neuron, therefore, requiring a physical area that scales with the number of synapses and neurons in the network. This was a serious limiting factor with older VLSI fabrication processes. However, technology scaling pushed by Moore’s law, and the emergence of novel nanoscale memory device technologies that can be used to implement synaptic and neural dynamics [5], [6], bring renewed interest to this approach and make it very competitive compared to the classical one of resorting to more compact digital designs based on shared time-multiplexed circuits, as was done, for example, for the IBM TrueNorth [16] and Intel Loihi [19] neuromorphic processors.

A clear difference between the use of pure digital circuits versus mixed-signal analog/digital ones is the higher sensitivity to noise and variability of the latter. The higher robustness of digital circuits is given by their feature of restoring signals to high or low states at each processing stage. This ensures accurate representation of signals and is a crucial requirement for deterministic and reproducible computation using the standard computing paradigms based on Boolean logic. However, animal brains are an existence proof that noisy and analog computational elements can, indeed, be used to perform reliable computation in sensory processing tasks at much lower power. Within this context, mixed-signal neuromorphic processors represent a promising approach that can potentially lead to robust computation by using the same strategies used by real nervous systems, such as adaptation, learning, and smoothing over space (for example, with population coding [26]) and time (through the low-pass filtering property of transferring the function of the physical computing elements).

Natural time in hardware neural processing systems

For the approach based on parallel instances of mixed analog/digital circuits as described previously, the most power-efficient way of processing time-varying signals is to use circuit time constants that are well matched to those of the dynamics of the signals that need to be processed. For example, real-world “natural” events and signals, such as speech, biosignals measured from the body, human gestures, and motor and sensory signals measured from roving robots, would require the synapse and neural circuits to have time constants in the range of 5–500 ms. It is important to realize that although the speed of the individual synapse or neuron computing elements in such architectures is very slow (for example, compared to digital circuits), the response time of the overall system can be extremely fast. This is due to the fact that the distributed parallel processing nodes in the system will be affected by device mismatch and have different initial conditions; so, upon the arrival of an external input, there will be many neurons very close to their firing threshold, and they will produce an output with a latency that is much shorter than their natural integration time constant. While these fast reaction times can be achieved with any (mismatched and matched) system, for example, via stochastic resonance, neural

Table 1. The quantitative comparison of mixed-signal neuromorphic processor specifications.

| | LANN-21 [8] | F-LANN [9] | LENA-IC [10] | IFAT [11] | NeuroGrid [12] | ROLLS [14] | DYNAP-SE [13] |
|-------------------------------|--------------------|----------------------|--------------------|---------------------|---------------------|----------------------|---------------------|
| Technology | 0.6 μm | 0.6 μm | 0.35 μm | 90 nm | 0.18 μm | 0.18 μm | 0.18 μm |
| Supply Voltage | 3.3 V | 3.3 V | 2.4 V | 1.2 V | 3 V | 1.8 V | 1.8 V |
| Core Area | 10 mm ² | 68.9 mm ² | 25 mm ² | 139 μm^2 | 170 mm ² | 51.4 mm ² | 7.5 mm ² |
| Neurons/Core | 21 | 128 | 100 | 2,000 | 65,636 | 256 | 256 |
| Synapses/Core | 129 | 16,384 | 30,000 | N/A | 4,096 | 128,000 | 16,000 |
| Fan-In/Fan-Out | 21/21 | 128/128 | 100/100 | N/A | N/A | 256/256 | 64/4,000 |
| Synaptic Weight | Capacitor | Capacitor | >10 bits | 8 bits | 13-bit shared | Capacitor | 1 + 1 bit |
| Online Learning | STRDP | STRDP | STDP | No | No | STRDP | No |
| Neurons/mm² | 174 | N/A | 27 | 7,142 | 360 | 1,089 | 812 |
| Energy per SOP | N/A | N/A | 10 pJ | 22 pJ | 31.2 pJ | 77 fJ | 17 pJ |

fJ: femtojoule; pJ: picojoule.

networks that use dedicated circuits per neuron and that are affected by mismatch do not need to implement explicit random number generators and cycle through each time-multiplexed neuron to inject the noise needed to achieve this effect.

The relatively long time constants required by this approach are not easy to realize using analog CMOS technology. Standard analog circuit design techniques either lead to bulky and silicon-area expensive solutions or fail to meet this condition, resorting to modeling neural dynamics at accelerated timescales [27]. One elegant solution to this problem is to combine the use of subthreshold circuit design techniques [3] with a current-mode design [28]. A very compact subthreshold log-domain circuit that can reproduce biologically plausible synaptic and neural temporal dynamics and that has been used in a wide variety of neuromorphic applications [25], [29]–[31] is the differential pair integrator (DPI) circuit [32], depicted in Figure 1. The analytic transfer function of this circuit can be derived following a translinear-loop log-domain analysis [2], [28]. From Figure 1(b), it follows that

$$\tau \left(1 + \frac{I_g}{I_{out}} \right) \frac{d}{dt} I_{out} + I_{out} = \frac{I_g I_{in}}{I_\tau} - I_g, \quad (1)$$

where I_{out} represents the synaptic or neuron dynamic variable, I_{in} the input signal, I_τ a user-defined leakage current, and I_g a global gain term useful, for example, for modeling spike-based learning, intrinsic plasticity, and synaptic scaling. This is a nonlinear differential equation. However, under the reasonable assumptions of nonnegligible input currents, such that $I_{in} \gg I_\tau$, and observing that, for such conditions, the output current I_{out} will eventually grow to be $I_{out} \gg I_g$, this equation simplifies to

$$\tau \frac{d}{dt} I_{out} + I_{out} = \frac{I_g}{I_\tau} I_{in}. \quad (2)$$

The circuit time constant is defined at $\tau \triangleq CU_T/\kappa I_\tau$. Long time constants are achieved by using a combination of large capacitors and small currents. Even though it is possible to implement such long time constants without sacrificing large amounts of silicon real estate area, for example, using high dielectric-constant materials, memristive devices, and active circuits that minimize leakage currents [31], there is a limit to the maximum time constants that can be implemented at the level of the single neural or synaptic components, and to the temporal scales that they can deal with for processing slow-changing signals. This is a problem that biological neural processing systems also face and that can be solved by using network dynamics to model attractors and long-term memory phenomena.

Signal processing in neuromorphic hardware on behavioral timescales

Biological nervous systems are capable of controlling behavior and processing sensory inputs in an adaptive and robust manner over a multitude of timescales that extend well beyond those of the single synapse and neuron time constants [33]. Indeed, in such systems, there is a multitude of neural circuits and mechanisms that underlies the ability to process and generate

temporal patterns over behavioral timescales [34]. A common circuit motive found throughout multiple parts of the cortex that is believed to subserve these functions is the “winner take all” (WTA) network [35], [36], which is a specific type of attractor network [37]. Theoretical studies have shown that such networks provide elementary units of computation that can stabilize and denoise the neuronal dynamics [35], [38], [39]. These theoretical considerations have been validated in neuromorphic hardware systems to generate robust behavior in closed sensorimotor loops [40]. However, to extend these models and hardware neural processing circuits to more complex systems, such as autonomous agents that can make decisions and generate goal-directed behavior, it is necessary to develop higher-level control strategies and theoretical frameworks compatible with mixed signal neuromorphic hardware, and endowing neuromorphic architectures with compositionality and modularity. The core challenge is to design neuronal networks for neuromorphic hardware that can create and stabilize a neuronal state that can, for example, drive movements of a robot that unfold at arbitrary timescales.

Recurrently connected neural populations can, indeed, create sustained activation to keep a neuronal state active for macroscopic, behavioral time intervals, providing a model of working memory [41]. Such sustained neuronal states do not need to lead to static behavior but should create an attractor in the behavioral space that generates the desired behavior. This desired behavior can be periodic or correspond to a fixed-point attractor [42].

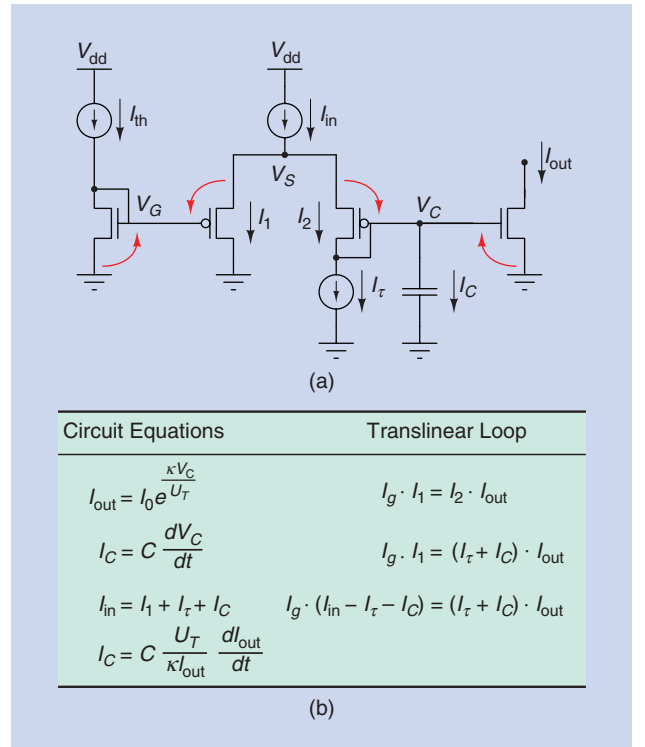


FIGURE 1. (a) The DPI circuit diagram; the red arrows show the translinear loop considered for the log-domain analysis. (b) The circuit and translinear loop equations used to derive the circuit transfer function. The term I_0 in the equation represents the transistor dark current, U_T the thermal voltage, and κ the subthreshold slope factor [3].

These attractor-based representations can sustain a variable duration of actions and perceptual states and help to bridge the neuronal and behavioral timescales in a robust way [23], [41]. The recurrent neural network dynamics that can be used for this purpose are described by the following equation [22]:

$$\tau \frac{d}{dt} u(x, t) = -u(x, t) + h + I(x, t) + \int f(u(x', t)) w(|x - x'|) dx'. \quad (3)$$

Here, $u(x, t)$ is a neuronal activation function defined across a perceptual or motor space, x , that is encoded by a neuronal population. Such a dimension, x , could represent a continuous sensory feature, such as color or orientation of a visual stimulus, or a motor variable, such as a hand position or location in space. The term h in (3) is a negative resting level of the neuronal population; the term $I(x, t)$ is an external input, and $f(\cdot)$ is a sigmoid nonlinearity that smoothly filters population activity. The final integral term in the equation expresses the lateral connectivity in the neuronal population with activity described by the activation function $u(x, t)$. In particular, the integral is a convolution shaped by the interaction kernel $w(|x - x'|)$ that only depends on the distance between two positions. The interaction kernel has a “Mexican hat” form of the type

$$w(|x - x'|) = A_{\text{exc}} e^{-\frac{(x-x')^2}{2\sigma_{\text{exc}}^2}} - A_{\text{inh}} e^{-\frac{(x-x')^2}{2\sigma_{\text{inh}}^2}}, \quad (4)$$

where $A_{\text{exc/inh}}$ and $\sigma_{\text{exc/inh}}$ are the amplitude and spread of the excitatory and inhibitory parts of the kernel. Such a pattern of lateral connections creates a soft WTA dynamic in the neuronal population: If they are supported by neighboring (in the behavioral space) neurons, the neurons that get activated first stay activated and inhibit other neurons in the population. This formalization of the dynamics of recurrent neural populations is known as a dynamic neural field (DNF) [43], [44]. Through decades, DNF theory was developed into a framework for a neuronal basis of cognition [22] that has been recently applied to the control of cognitive robotic agents [23], [45].

The DNF dynamics described by (3) can be simulated on a computer using the Euler or Runge-Kutta method for the numerical integration of the integro-differential equation. Such simulations were developed in the past using MATLAB (<https://dynamicfieldtheory.org/cosivina/>) and C++ [46] (<https://cedar.ini.rub.de>) software libraries, and they usually require substantial computing power since the dense recurrent connections in large neuronal fields do not allow for parallelizing architectures efficiently. In some cases, when interaction kernels are separable and unchanging, the computation can be brought into Fourier space and run more efficiently. However, even in these cases, the performance of large-scale DNF architectures on a conventional computer is still unsatisfactory for real-world robotic applications. To the contrary, implementation of DNFs in neuromorphic hardware enables running these networks in real time, since computation is performed in parallel using hardware connections instead of simulating convolutions and numerical integration over time.

Figure 2(a) shows a scheme of a WTA/DNF implementation with spiking neurons. Here, red circles designate neurons, a larger number of which form an excitatory pool that represents the behavioral variable; the smaller pool of neurons forms an inhibitory group that realizes the inhibitory part of interaction kernel in (3). Red lines are excitatory, and blue lines are inhibitory connections (shown for one of the inhibitory neurons each).

The stable attractors created by such WTA dynamics are critical to enable behavior learning in a closed sensory-motor loop in which the sensory input changes continually as the agent generates action. To learn a mapping between a sensory state and its consequences, or a precondition and an action, the sensory state, before the action, needs to be stored in a neuronal representation. This can be achieved by creating a reverberating activation in the neuronal population that can be sustained for the duration of the action, even if the initial input ceases. The sustained activity can be used to update sensorimotor mappings when a rewarding or punishing signal is obtained [45], [47].

Implementing on-chip learning in neuromorphic processors

Learning is probably the most important aspect of neural processing systems: It allows us to train ANNs to perform pattern classification and recognition tasks, and in biological neural systems, it allows animals to form memories and create associations. Most importantly, however, it endows agents with the capability to adapt to changes in the statistics of the input signals or changes in the properties of their actuators across different timescales.

Usually, in machine learning, the supervised-, unsupervised-, and reinforcement-learning procedures are distinguished. All three cases of learning must be addressed, both in biological neural systems and on neuromorphic hardware that emulates those principles, using spiking neuronal dynamics and spike-based learning rules [48], [49]. One line of research, thus, tries to adapt the learning methods and training procedures developed for “rate-based,” continuous-valued ANNs to spiking neural networks, which do not support nonlocal computing across synaptic weights or the computation of derivatives [50]–[52]. The hardware principles that we present here will support these developments. In our own work, additionally, we aim to develop learning architectures that enable the adaptation of synaptic weights and other network parameters when the neuronal network is used to produce behavior. Such “online” continual learning relies on stabilized representations of behavioral states (that include both perceptual states and action intentions) and their consequences or preconditions in attractor networks, presented in the “Signal Processing in Neuromorphic Hardware on Behavioral Timescales” section. The weight update mechanism itself is not critical and can, then, be implemented by simple Hebbian learning mechanisms.

In artificial neural processing systems, learning typically involves the update of synaptic weight values. In hardware, this can require significant amounts of extra resources: In time-multiplexed systems, these resources are typically in the form of state memory, memory-transfer bandwidth, and power. These

extra resources can be especially significant if the storage is done using memory banks that are not on the processor die (for example, dynamic random-access memory in large-scale systems). On the other hand, the overhead needed to implement learning in mixed-signal neuromorphic architectures that place multiple instances of synaptic circuits per emulated synapse is very small. For example, Figure 3, depicting the chip micrograph of the ROLLS device [14], shows how the synapses that comprise learning algorithms occupy approximately the same area of the nonplastic synapses that have fixed programmable weights and short-term plasticity (STP) dynamics. Since each of these synapse circuits can be stimulated individually by the chip input spikes, they can operate in parallel using slow dynamics (for example, driven by picoampere currents), without having to transfer state memory at high speeds to external memory banks. Here, the area used by the parallel circuits enables us to save bandwidth and power in implementing neural dynamics. Furthermore, the feature of implementing biologically plausible time constants and making use of explicit natural timescales enables us to use the fast digital AER [7] circuits for stimulating multiple synapses in parallel.

In our work, we used the ROLLS device in Figure 3 to enable learning in WTA networks to implement a hardware model of sequence learning [53] as well as learning maps [54] and sensorimotor mappings [55] in neuromorphic agents. Specifically, we used interconnected populations of spiking neurons in a WTA fashion, creating excitatory recurrent connections between

neurons that encode similar features. When a group of such interconnected neurons is activated, the excitatory connections create an activation flow between neurons that can sustain itself after the initiating input yields. Such sustained activation forms an attractor that can drive down-stream structures, resulting in a movement of the agent; the time structure of this movement can be decoupled from the dynamics and timing of sensory input. Moreover, learning between sustained attractor states can be triggered when a rewarding or error signal is perceived.

The rather dense WTA connectivity requires parallel processing to be efficiently computed. Moreover, nonlinearities of the DNF dynamics, which are crucial for its attractor properties, need to be processed for a large number of neurons in parallel. The dedicated neuronal circuits in neuromorphic devices, such as the one in Figure 3, lead to a direct implementation of the neuronal attractor networks in the wiring on the chip, with local synapses and neurons performing signal processing (such as filtering and smoothing), symbolic computation (selecting stable attractors), and state-holding (memory storage). Such implementation is more efficient compared to hardware with time multiplexing and virtual time, in terms of energy, because it does not require moving the state data (neuronal activation and weight values) between separate memory blocks and processing ones. We, thus, argue that neuromorphic devices that feature real-time processing with timescales matching the task, and with a massively parallel network of analog computing elements, when matched with

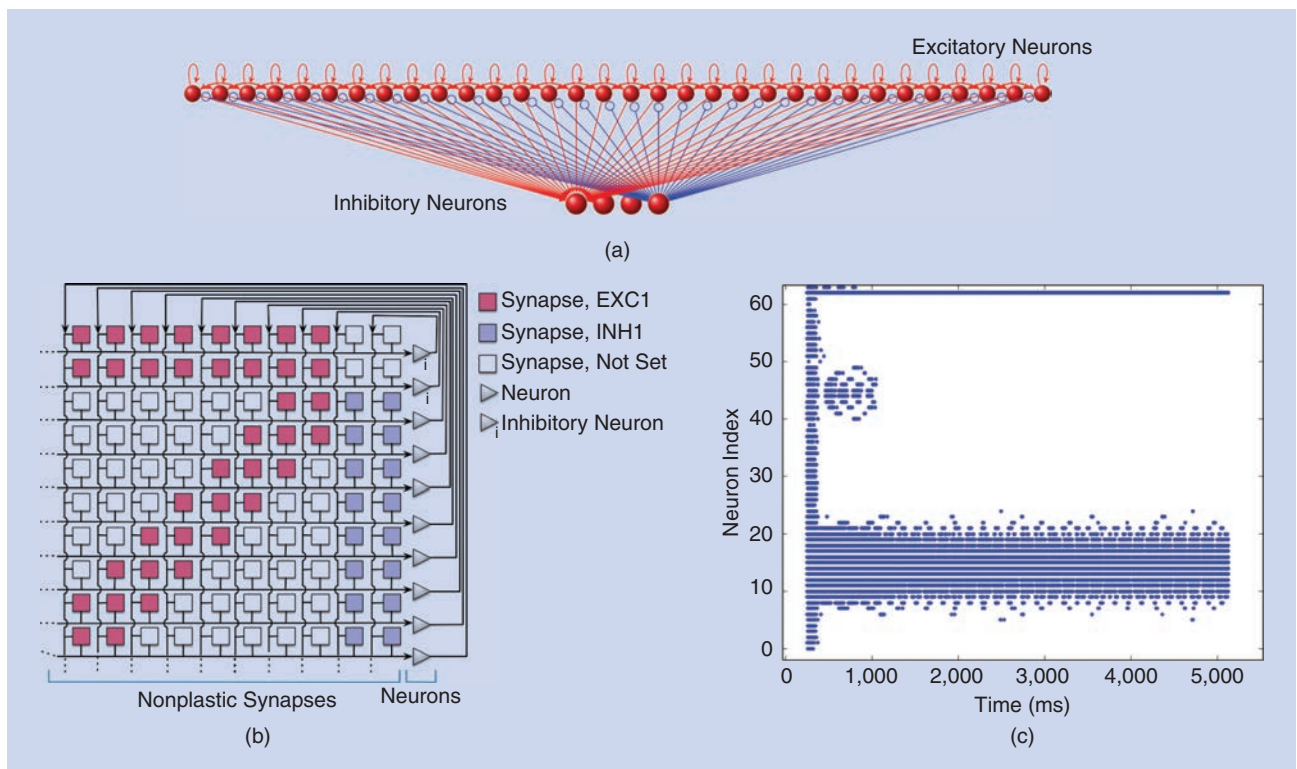


FIGURE 2. (a) The neuronal connectivity realizing a WTA/DNF (red are excitatory connections and blue are inhibitory connections; only one set of excitatory-to-inhibitory and inhibitory-to-excitatory connections is shown). (b) The same connectivity is realized in nonplastic synapses on a mixed-signal neuromorphic processor. (c) The illustration of the noise-reducing properties of the DNF dynamics: a raster plot of spiking activity in a WTA architecture implemented on a neuromorphic processor [14]. The distractor input is suppressed after 1 s.

a neuronal architecture of attractor-based population dynamics, can lead to efficient implementations of neuromorphic systems capable of generating cognitive behavior. In the following sections, we present representative examples of such neuromorphic agents.

Signal processing for neuromorphic agents

In this section, we describe neuronal architectures that make use of the properties of neuromorphic devices with explicit representation of neuronal substrate and real time, using concepts of attractor dynamics in neuronal populations as a means to program the devices to implement desired functionalities. We describe, in detail, two seminal architectures: a realization of a closed-loop navigation controller inspired by a Braitenberg vehicle, and a sequence-learning architecture that makes use of plastic on-chip synapses to store sequences of observations. We review a few more examples where on-chip plasticity was used to learn along with behavior. Both these architectures were realized on the ROLLS neuromorphic processor [14]. Despite the small size of the network that was implemented, the neuromorphic processor is capable of producing robust behavior on a robotic agent in real time.

Closed-loop sensorimotor control

The first neuromorphic architecture demonstrates how the behavior of a simple roving agent can be controlled by a spiking neuronal network. Such behavior can be generated by a very

simple neuronal system as has been demonstrated by V. Braitenberg with his classical “vehicles” that generate meaningful behavior in a structured environment based on the simple wiring of their sensors and motors [56]. According to this view, the most basic ability required to generate goal-directed behavior is the capacity to differentiate between different sensory inputs. In simple terms, the agent must be able to tell one sensory state from another one. In terms of the neuronal architecture, this means that the system needs to represent combinations of visual features and their spatial locations to select the spatial target for a movement. Note that such combinations are also generated in deep convolutional neural network architectures in different feature maps. The spatial component, however, typically gets lost across layers of a network that is trained for a recognition task. The first part of our architecture achieves such differentiation or representation of sensory states (note, we are showing a small-scale example, here, on a chip with 256 spiking neurons).

Figure 4 presents a neuronal architecture that realizes one of Braitenberg’s controllers on the neuromorphic device ROLLS. We use a WTA network to represent visual input obtained by a neuromorphic, event-based camera DVS mounted on a miniature robotic vehicle, Pushbot. In particular, neurons in two WTAs receive an external spike for each event in their receptive field: The first WTA population (“Target” in the figure) has weak lateral interactions and can represent several potential targets in the field of view of the robot. The second

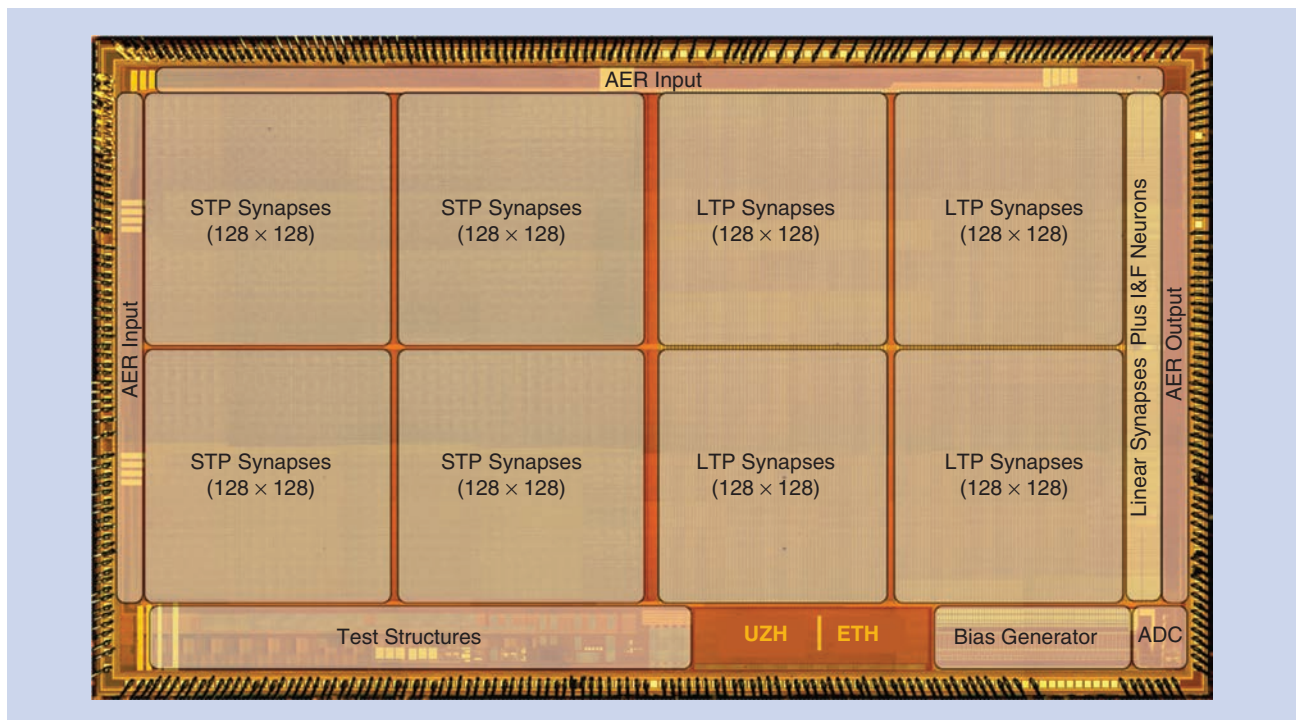


FIGURE 3. The chip micrograph of the ROLLS neuromorphic processor. Silicon-neuron integrate and I&F circuits are placed at the right of the die area and connected to an array of 256×512 synapses. Half of the synapse array is composed of learning synapses that exhibit long-term plasticity (LTP), and the other half is composed of fixed-weight synapses that exhibit STP. Input and output spikes are routed to the synapses from the neurons via asynchronous digital address-event representation (AER) circuits. On-chip DACs (bias generators) can be programmed to set the bias values of the analog synapse and neuron circuits. On-chip analog-to-digital converters (ADCs) are used to read out the currents from the current-mode DPI circuits that emulate synaptic dynamics. UZH: Universität Zürich; ETH: Eidgenössische Technische Hochschule Zürich.

representation, each active neuron signals that there is some input coming from the portion of the sensor that it observes. If several objects are present in front of a camera, several neurons will be active. In the selective target population, moreover, the WTA connectivity makes sure that a single target location is selected and that distractors are suppressed [57]–[59], [74].

On the motor side, the neurons that initiate the turning movement of the robot are connected to the obstacle left/right populations. These motor neurons, if active, cause the robot to turn either to the right or left. Thus, each of the two motor populations in Figure 4 activates a motor primitive that causes the robot's movement. In particular, the rotational velocity of the robot is set proportionally to the firing rate of the respective (left or right) motor population. The presence of an obstacle also makes the robot slow down; activity in the "Obstacles" populations inhibits the "Speed" neuronal population. The firing rate of this latter population sets the robot's forward speed.

Overall, the simple neuronal architecture in Figure 4 enables the robot to drive around, avoiding obstacles and approaching targets [57]–[59], [74]. This behavior is produced by a dynamical system created by the sensorimotor "embodiment" of the robot that is situated in an environment, and the neuronal system that closes the behavior loop, connecting perception to motor control. This controller does not include any algorithmic arbitration: The sensory signals (DVS events and gyroscope measurements) directly drive neurons and, eventually, the motors. The neuronal architecture ensures that the correct behavior is produced. The fact that the neurons on the ROLLS chip are instantiated physically makes such efficient, direct connection possible, reducing demands on the memory and arbitration that are usually managed by the CPU.

The neuronal architecture presented here does not include any learning. Indeed, this architecture is wired for a particular task, although the precise neuronal parameters can be found by a machine-learning procedure [60] instead of human labor tuning. In the next section, we present a neuronal architecture on the ROLLS chip that includes learning using plastic on-chip synapses. This architecture is one of our recent examples that shows how representations—of temporal or spatial behaviorally-relevant patterns, such as sequences or maps—can be learned using principles of attractor dynamics in hardware with explicit, physical representation of space and time.

Learning and processing sequences in the neuromorphic device

Recently, we implemented the basic serial-order sequence-learning model based on DNFs [61] on the ROLLS chip [53]. In this architecture, a WTA population represents the content of the items in a sequence (here, the location of a target on the horizontal axis of the DVS's FoV). Other neuronal populations represent serial order positions along the sequence: the first, second, and third ordinal group, and so on. Each group of these ordinal neurons is connected to the content WTA with plastic synapses that are updated when an ordinal neuron and a

WTA neuron are active at the same time, according to a local synaptic plasticity rule. Groups of memory neurons, driven by the ordinal neurons, create sustained activation and keep track of the position in the sequence during periods of sequential transitions, when ordinal groups are inhibited. This inhibition is induced by a condition-of-satisfaction (CoS) system: a group of neurons that is activated when an item in the sequence has been successfully executed [61], [62].

In [54], we show how the sequence of states can be learned and replayed by a robotic agent, whose sensors and motors are interfaced to the neuromorphic device implementing the sequence learning architecture in Figure 5. In this example, again, the physical identity of neurons is important to be able to efficiently, directly distribute activity patterns in the neuronal architecture. Moreover, the ability of the attractor networks to bridge different lengths of time is critical here: Note that during sequence learning and production, each item in a sequence can take different—and unknown in advance—time intervals. Neuronal dynamics can sustain these long time intervals because stable activity bumps are created in the WTA neuronal population and in the ordinal groups. No clock is needed to keep track of time in these neuronal systems explicitly, although storing typical durations of actions is also possible in the neuronal population dynamics [63].

Another example of a neurodynamic architecture built using attractor networks and allocating dedicated circuits for each synapse/neuron in the network is the neuromorphic self-localization and map-formation architecture presented in [54], [64] and [65]. In this system, first, a WTA neuronal population keeps track of the correct heading of the agent, performing path integration based on the motor commands sent to the robot. Another neuronal population keeps track of the position of the robot in an environment. The plastic synapses of the ROLLS chip are used to learn a simple map of the environment as the robot navigates it and senses walls with a bumper. The map is effectively an association between the representation of position (and orientation) of the robot and a neuronal representation of collisions. Learning happens in the moment when the event induced by the collision with an object activates the "Collision" population of neurons. Coactivation of these neurons and neurons that represent the current position in real time leads to an increase of the plastic synapses between these two groups, forming a collisions map as the robot explores the environment. When the robot revisits the place of a previously experienced collision, but no collision happens there, the "Collision" population will be activated through the plastic synapses, anticipating a collision. However, the firing rate of this population will be lower than during an actual collision, when neurons are stimulated by the sensory input. This causes synaptic depression in the plastic synapses and leads to a gradual forgetting of the associations that are no longer valid.

The same principles of explicit space and time representation have been used to develop a neuromorphic proportional-integral (PI) controller that triggers the learning of a mapping from the target speed of a robot to a motor command that

realizes this speed [55]. Such mapping is an instantiation of a simple inverse model, which is learned using plastic synapses on the ROLLS chip. Here, learning is triggered when the PI controller (realized using the same principles of population dynamics and attractor WTA networks) converges and activates a zero-error neuronal population. Again, in this architecture, sensory inputs drive neuronal representations in real time, and learning is activated autonomously.

In all these neuromorphic architectures, a neuronal system is designed that connects sensors and motors to solve a particu-

lar task. Learning is reserved for certain portions of these architectures; for example, a mapping between a sensory and motor space, or between a representation of the ordinal position and perceptual content. The WTA connectivity of neuronal populations that represent perceptual states ensures that representations are localized and stabilized, limiting learning in time and in space. We have developed more architectures that include learning; for example, in [64], we show how on-chip plasticity combined with a WTA structure can also be used to learn to differentiate patterns in an unsupervised manner, while [55]

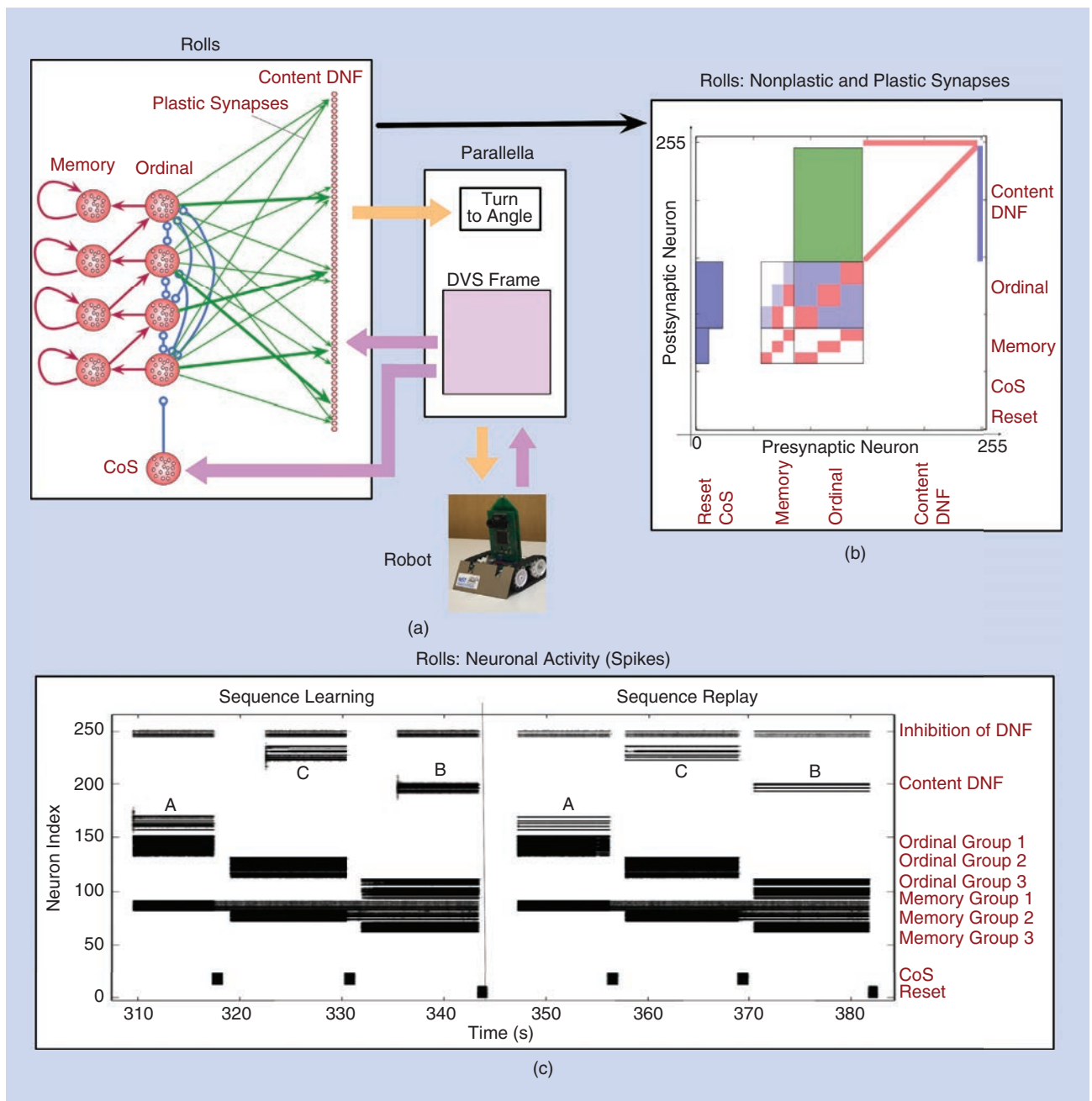


FIGURE 5. (a) The schematics of the architecture for sequence learning on the ROLLS chip. (b) Both plastic and non-plastic synapses. Plastic synapses are in the green block, and non-plastic synapses are red (excitatory) or blue (inhibitory). (c) A raster plot of neuronal activity in a sequence-learning experiment in which the robot observes three inputs, stores their order in plastic on-chip synapses, and replays the items. See [54] and the main text for details.

and [59] show how neuronal populations can be used to perform arithmetic operations, such as the summation and subtraction required, for example, to realize a PI controller in a neural network, or to perform reference frame transformations [59]. The latter work emphasizes that some computation that can easily be solved on a conventional von Neumann computing platform might require rethinking and the design of a whole neuronal architecture in neuromorphic computing devices.

For any given task that we have solved using neuromorphic computing and discussed here, a conventional software solution can be developed. Indeed, the digital computing architecture can be used to realize any computation, since it is Turing-complete. There are two main reasons why we believe that developing neuromorphic agents is important, nevertheless. First, the cost of computation becomes a nonnegligible issue as computing is scaled up in terms of the amount of sensory information that needs to be processed in real time on a compact device with limited power. Thus, even if more precise digital solutions exist for the problems of map formation, sequence learning, and motor control, and that employ digital computers and conventional software, truly neuromorphic solutions that use physics to realize neuronal computing can solve these tasks using several orders of magnitude less power. Of course, due to today's limitations in the number of artificial neurons realized in hardware, the resulting neuronal solutions provide a lower precision, which, however, is sufficient for many tasks that include "soft" actuators and imprecise sensors, in particular where closed-loop control can be used to keep the behavior at a tolerable accuracy.

The second reason lies in the homogeneity and adaptivity of the neuromorphic computing architectures. Indeed, all architectures that we briefly reviewed here use just a handful of neuronal structures—WTAs and plastic connections between them—to realize many different functions: representation, feature binding, attention, memory formation, decision making, reference frame transformation, and so forth. Indeed, in the theoretical framework of DNFs, it has been shown that most cognitive processing can be done with similar building blocks that realize attractor dynamics in neuronal populations [22]. Augmented with simple Hebbian learning, intrinsic plasticity, and homeostasis, such neuronal architectures can autonomously adapt to new situations and be easily "reprogrammed" to produce new behaviors just by using demonstration or self-generated supervision signals [66]–[69]. Combined with the ability to represent continuous variables in an adaptive, task-dependent way, neuromorphic computing steps outside the Turing computing framework and, as the hardware matures and supports larger networks, could lead to more scalable and robust solutions to tasks that involve interaction with the real world.

Conclusions

Neuromorphic computing represents a promising paradigm for artificial cognitive systems that may become an enabling element for the design of power-efficient, real-time, and compact solutions for electronic devices that process sensory data and generate behavior in dynamic, real-world environments.

To minimize power consumption, it is important to match the dynamics of neuromorphic computing elements to the timescale of the physical processes that drive them and that the devices control. The impact of this approach is significant because, in addition to autonomous robots, "cognitive agents" represent low-power, always-on embedded sensory-computing systems (for example, in cell phones or wearable devices), intelligent wireless sensors (for Internet of Things applications), and intelligent microbiosensors (for personalized medicine, brain-machine interfaces, and prosthetics), and other types of microscale autonomous devices that extract structure from the data they acquire through their interactions with the environment and make decisions on how and when to act (to transmit information, power-up further processing stages, and so forth).

Although many of the organizing principles of nervous systems have successfully been adopted for the design of artificial intelligence systems, two of the most powerful ones (that is, that of letting time represent itself, and that of using the physical instantiations of parallel circuits rather than single time-multiplexed ones) are not yet fully exploited. Here, we presented examples of systems that implement these principles and described a computational approach that extends the temporal processing abilities of such devices to arbitrary timescales using attractor dynamics and the DNF framework. We discussed several examples of neurodynamic architectures that make use of the physical instantiation of spiking neurons in hardware and their real-time dynamics, matched to biological timescales. We presented architectures that were designed based on the available sensory and motor systems and the given task as spiking neural networks that can be realized in neuromorphic hardware. In some cases, when such spiking neural network implementation is not straightforward, other methods can be used to either learn the architecture from examples of the desired behavior, recorded with a teleoperated sensorimotor plant using, for example, deep neural networks, or to approximate the dynamical system of the neuronal controller with spiking neurons using the neural engineering framework [70].

Many more examples are being proposed for using dedicated circuits for neurons and synapses to take advantage of emerging memory technologies based on memristive devices [71]. These novel types of architectures differ from the more classical type of deep network or convolutional network accelerators mainly for breaking the von Neumann bottleneck [72] by having distributed memory elements that act also as computing devices. This in-memory computing approach enables dramatic power savings. For example, the (pure CMOS-based) in-memory computing DYNAP-SE processor presented in Table 1 is at least a factor of 100 times more power efficient than Spinnaker2 (for example, compare the power consumption figures in [13] versus [73]), which represents one of the most recent state-of-the-art spiking neural network accelerators implemented following the classical synchronous logic, time-multiplexed, microprocessor approach.

We believe the examples presented in this article represent the first steps toward the realization of power-efficient neuromorphic systems and robust computing architectures

optimally suited for tasks in which behavior needs to be generated by autonomous neuromorphic agents in real time, while staying adaptive in complex environments.

Acknowledgments

Many of the concepts presented here were inspired by the discussions held at the CapoCaccia Cognitive Neuromorphic Engineering Workshop. The quantitative comparison in Table 1 was performed by Mohammad Ali Sharif Shazileh. This work received funding from the European Research Council under grant 724295 (Neuro Agents) and the Swiss National Science Foundation project Ambizione (grant PZ00P2_168183_1).

Authors

Giacomo Indiveri (giacomo@ini.uzh.ch) received his M.Sc. degree in electrical engineering and Ph.D. degree in computer science from the University of Genoa, Italy, and his habilitation degree on neuromorphic engineering at ETH Zürich in 2006. He is the director of the Institute of Neuroinformatics of the University of Zürich and ETH Zürich, and he holds a professor position at the University of Zürich, Switzerland. He is a recipient of two European Research Council (ERC) grants: he was awarded an ERC Starting Grant in 2011 and an ERC Consolidator Grant in 2017.

Yulia Sandamirskaya (ysandamirskaya@ini.uzh.ch) received a degree in physics from Belarusian State University, Minsk, and a Dr. rer. nat. degree from the Institute for Neural Computation at the Ruhr-Universität Bochum, Germany. She is a group leader in the Institute of Neuroinformatics of the University of Zürich and ETH Zürich. Her group, Neuromorphic Cognitive Robots, develops neurodynamic architectures for embodied cognitive agents. In particular, she studies memory formation, motor control, and autonomous learning in spiking and continuous neural networks, realized in neuromorphic hardware interfaced to robotic sensors and motors. She is the chair of the European Society for Cognitive Systems and the coordinator of the NEUROTECH project that supports and develops the neuromorphic computing community in Europe.

References

- [1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. doi: 10.1038/nature14539.
- [2] E. Chicca, F. Stefanini, C. Bartolozzi, and G. Indiveri, "Neuromorphic electronic circuits for building autonomous cognitive systems," *Proc. IEEE*, vol. 102, no. 9, pp. 1367–1388, Sept. 2014. doi: 10.1109/JPROC.2014.2313954.
- [3] C. Mead, "Neuromorphic electronic systems," *Proc. IEEE*, vol. 78, no. 10, pp. 1629–1636, Oct. 1990. doi: 10.1109/5.58356.
- [4] G. Indiveri, B. Linares-Barranco, T. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek et al., "Neuromorphic silicon neuron circuits," *Front. Neurosci.*, vol. 5, pp. 1–23, May 2011. doi: 10.3389/fnins.2011.00073.
- [5] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano Lett.*, vol. 10, no. 4, pp. 1297–1301, Mar. 2010. doi: 10.1021/nl904092h.
- [6] S. Saighi, C. Mayr, B. Linares-Barranco, T. Serrano-Gotarredona, H. Schmidt, G. Lecerf, J. Tomas, J. Grollier et al., "Plasticity in memristive devices," *Front. Neurosci.*, vol. 9, no. 51, Mar. 2015. doi: 10.3389/fnins.2015.00051.
- [7] S.-C. Liu, T. Delbruck, G. Indiveri, A. Whatley, and R. Douglas, *Event-Based Neuromorphic Systems*. Hoboken, NJ: Wiley, 2014.

- [8] E. Chicca, D. Badoni, V. Dante, M. D'Andreagiovanni, G. Salina, L. Carota, S. Fusi, and P. Del Giudice, "A VLSI recurrent network of integrate-and-fire neurons connected by plastic synapses with long term memory," *IEEE Trans. Neural Netw.*, vol. 14, no. 5, pp. 1297–1307, Sept. 2003. doi: 10.1109/TNN.2003.816367.
- [9] M. Giulioni, P. Camilleri, V. Dante, D. Badoni, G. Indiveri, J. Braun, and P. Del Giudice, "A VLSI network of spiking neurons with plastic fully configurable 'stop-learning' synapses," in *Proc. 2008 15th IEEE Int. Conf. Electronics, Circuits, and Systems*, pp. 678–681. doi: 10.1109/ICECS.2008.4674944.
- [10] S. Brink, S. Nease, P. Hasler, S. Ramakrishnan, R. Wunderlich, A. Basu, and B. Degnan, "A learning-enabled neuron array IC based upon transistor channel models of biological phenomena," *IEEE Trans. Biomed. Circuits Syst.*, vol. 7, no. 1, pp. 71–81, Feb. 2013. doi: 10.1109/TBCAS.2012.2197858.
- [11] J. Park, S. Ha, T. Yu, E. Neftci, and G. Cauwenberghs, "A 65k-neuron 73-meV/s 22-pJ/event asynchronous micro-pipelined integrate-and-fire array transceiver," in *Proc. 2014 IEEE Biomedical Circuits and Systems Conf. (BioCAS)*, pp. 675–678. doi: 10.1109/BioCAS.2014.6981816.
- [12] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J. Bussat, R. Alvarez-Icaza, J. Arthur et al., "Neurogrid: A mixed-analog-digital multi-chip system for large-scale neural simulations," *Proc. IEEE*, vol. 102, no. 5, pp. 699–716, Apr. 2014. doi: 10.1109/JPROC.2014.2313565.
- [13] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPS)," *IEEE Trans. Biomed. Circuits Syst.*, vol. 12, no. 1, pp. 106–122, Feb. 2018. doi: 10.1109/TBCAS.2017.2759700.
- [14] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumslawska, and G. Indiveri, "A re-configurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses," *Frontiers Neuroscience*, vol. 9, no. 141, pp. 1–17, Apr. 2015. doi: 10.3389/fnins.2015.00141.
- [15] J. Schemmel, D. Brüderle, A. Grubel, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *Proc. 2010 IEEE Int. Symp. Circuits and Systems*, pp. 1947–1950. doi: 10.1109/ISCAS.2010.5536970.
- [16] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam et al., "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Sci.*, vol. 345, no. 6197, pp. 668–673, Aug. 2014. doi: 10.1126/science.1254642.
- [17] S. Furber, F. Galluppi, S. Temple, and L. Plana, "The SpiNNaker project," *Proc. IEEE*, vol. 102, no. 5, pp. 652–665, May 2014. doi: 10.1109/JPROC.2014.2304638.
- [18] C. Frenkel, J. Legat, and D. Bol, "A 0.086-mm² 9.8-pJ/sop 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28nm CMOS," *IEEE Trans. Biomed. Circuits Syst.*, vol. 13, no. 1, pp. 145–158, Feb. 2019. doi: 10.1109/TBCAS.2018.2880425.
- [19] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi et al., "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan.-Feb. 2018. doi: 10.1109/MM.2018.112130359.
- [20] F. Conti and L. Benini, "A ultra-low-energy convolution engine for fast brain-inspired vision in multicore clusters," in *Proc. 2015 Design, Automation and Test Europe Conf. Exhibition*, pp. 683–688. doi: 10.7873/DATE.2015.0404.
- [21] A. Aimar, H. Mostafa, E. Calabrese, A. Rios-Navarro, R. Tapiador-Morales, I.-A. Lungu, M. B. Milde, F. Corradi et al., "Nullhop: A flexible convolutional neural network accelerator based on sparse representations of feature maps," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 3, pp. 644–656, Mar. 2019. doi: 10.1109/TNNLS.2018.2852335.
- [22] G. Schöner and J. Spencer, Eds., *Dynamic Thinking: A Primer on Dynamic Field Theory*. London: Oxford Univ. Press, 2015.
- [23] Y. Sandamirskaya, S. K. Zibner, S. Schneegans, and G. Schöner, "Using dynamic field theory to extend the embodiment stance toward higher cognition," *New Ideas Psychology*, vol. 31, no. 3, pp. 322–339, Dec. 2013. doi: 10.1016/j.newideapsych.2013.01.002.
- [24] M. R. Azghadi, N. Iannella, S. Al-Sarawi, G. Indiveri, and D. Abbott, "Spike-based synaptic plasticity in silicon: Design, implementation, application, and challenges," *Proc. IEEE*, vol. 102, no. 5, pp. 717–737, May 2014. doi: 10.1109/JPROC.2014.2314454.
- [25] S. Mitra, S. Fusi, and G. Indiveri, "Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI," *IEEE Trans. Biomed. Circuits Syst.*, vol. 3, no. 1, pp. 32–42, Feb. 2009. doi: 10.1109/TBCAS.2008.2005781.
- [26] W. Ma, J. Beck, P. Latham, and A. Pouget, "Bayesian inference with probabilistic population codes," *Nat. Neurosci.*, vol. 9, no. 11, pp. 1432–1438, Oct. 2006. doi: 10.1038/nn1790.
- [27] J. Schemmel, D. Brüderle, K. Meier, and B. Ostendorf, "Modeling synaptic plasticity within networks of highly accelerated I&F neurons," in *Proc. 2007 IEEE Int. Symp. Circuits and Systems*, pp. 3367–3370. doi: 10.1109/ISCAS.2007.378289.
- [28] C. Tomazou, F. Lidgey, and D. Haigh, Eds., *Analogue IC design: The Current-Mode Approach*. Stevenage, U.K.: Peregrinus, 1990.

- [29] T. Rost, H. Ramachandran, M. P. Nawrot, and E. Chicca, "A neuromorphic approach to auditory pattern recognition in cricket phonotaxis," in *Proc. 2013 European Conf. Circuit Theory and Design (ECCTD)*, pp. 1–4, doi: 10.1109/ECCTD.2013.6662247.
- [30] A. Banerjee, S. Kar, S. Roy, A. Bhaduri, and A. Basu, "A current-mode spiking neural classifier with lumped dendritic nonlinearity," in *Proc. 2015 IEEE Int. Symp. Circuits and Systems, (ISCAS)*, pp. 714–717, doi: 10.1109/ISCAS.2015.7168733.
- [31] N. Qiao, C. Bartolozzi, and G. Indiveri, "An ultralow leakage synaptic scaling homeostatic plasticity circuit with configurable time scales up to 100 ks," *IEEE Trans. Biomed. Circuits Syst.*, vol. 11, no. 6, pp. 1271–1277, Dec. 2017, doi: 10.1109/TBCAS.2017.2754383.
- [32] C. Bartolozzi and G. Indiveri, "Synaptic dynamics in analog VLSI," *Neural Comput.*, vol. 19, no. 10, pp. 2581–2603, Oct. 2007, doi: 10.1162/neco.2007.19.10.2581.
- [33] C. V. Buhusi and W. H. Meck, "What makes us tick? Functional and neural mechanisms of interval timing," *Nat. Rev. Neurosci.*, vol. 6, no. 10, pp. 755–765, Sept. 2005, doi: 10.1038/nrn1764.
- [34] J. J. Paton and D. V. Buonomano, "The neural basis of timing: Distributed mechanisms for diverse functions," *Neuron*, vol. 98, no. 4, pp. 687–705, May 2018, doi: 10.1016/j.neuron.2018.03.045.
- [35] R. J. Douglas and K. A. Martin, "Neuronal circuits of the neocortex," *Annu. Rev. Neurosci.*, vol. 27, pp. 419–451, July 2004, doi: 10.1146/annurev.neuro.27.070203.144152.
- [36] R. Douglas and K. Martin, "Recurrent neuronal circuits in the neocortex," *Curr. Biol.*, vol. 17, no. 13, pp. R496–R500, July 2007, doi: 10.1016/j.cub.2007.04.024.
- [37] N. Brunel, *Network Models of Memory*. Amsterdam, The Netherlands: Elsevier, 2004.
- [38] W. Maass, "On the computational power of winner-take-all," *Neural Computation*, vol. 12, no. 11, pp. 2519–2535, Nov. 2000.
- [39] U. Rutishauser, R. Douglas, and J. Slotine, "Collective stability of networks of winner-take-all circuits," *Neural Comput.*, vol. 23, no. 3, pp. 735–773, Feb. 2011, doi: 10.1162/NECO_a_00091.
- [40] E. Neftci, J. Binas, U. Rutishauser, E. Chicca, G. Indiveri, and R. Douglas, "Synthesizing cognition in neuromorphic electronic systems," *Proc. Nat. Academy Sci. United States America*, vol. 110, no. 37, pp. E3468–E3476, Sept. 2013, doi: 10.1073/pnas.1212083110.
- [41] J. S. Johnson, J. P. Spencer, and G. Schöner, "Moving to higher ground: The dynamic field theory and the dynamics of visual cognition," *New Ideas Psychology*, vol. 26, no. 2, pp. 227–251, Aug. 2008, doi: 10.1016/j.newideapsych.2007.07.007.
- [42] V. Mante, D. Sussillo, K. V. Shenoy, and W. T. Newsome, "Context-dependent computation by recurrent dynamics in prefrontal cortex," *Nature*, vol. 503, no. 7474, pp. 78–84, Nov. 2013, doi: 10.1038/nature12742.
- [43] S. Amari, "Dynamics of pattern formation in lateral-inhibition type neural fields," *Biological Cybern.*, vol. 27, no. 2, pp. 77–87, June 1977, doi: 10.1007/BF00337259.
- [44] H. Wilson and J. Cowan, "A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue," *Biological Cybern.*, vol. 13, no. 2, pp. 55–80, Sept. 1973, doi: 10.1007/BF00288786.
- [45] Y. Sandamirskaya, "Dynamic neural fields as a step toward cognitive neuromorphic architectures," *Front. Neurosci.*, vol. 7, pp. 276–289, Jan. 2014, doi: 10.3389/fnins.2013.00276.
- [46] O. Lomp, M. Richter, S. K. U. Zibner, and G. Schöner, "Developing dynamic field theory architectures for embodied cognitive systems with cedar," *Front. Neurobot.*, vol. 10, pp. 1–18, Nov. 2016, doi: 10.3389/fnbot.2016.00014.
- [47] Y. Sandamirskaya, J. Conradt, "Learning sensorimotor transformations with dynamic neural fields," in *Proc. ICANN 2013: Artificial Neural Networks and Machine Learning*, vol. 8131, pp. 248–255, New York: Springer.
- [48] R. Gütiğ and H. Sompolinsky, "The tempotron: A neuron that learns spike timing-based decisions," *Nat. Neurosci.*, vol. 9, no. 3, pp. 420–428, Feb. 2006, doi: 10.1038/nn1643.
- [49] W. Gerstner and W. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge, U.K.: Cambridge Univ. Press, 2002.
- [50] E. O. Neftci, C. Augustine, S. Paul, and G. Deterakis, "Event-driven random back-propagation: Enabling neuromorphic deep learning machines," *Front. Neurosci.*, vol. 11, pp. 1–18, June 2017, doi: 10.3389/fnins.2017.00324.
- [51] R. Urbanczik and W. Senn, "Learning by the dendritic prediction of somatic spiking," *Neuron*, vol. 81, no. 3, pp. 521–528, Feb. 2014, doi: 10.1016/j.neuron.2013.11.030.
- [52] J. C. Thiele, O. Bichler, and A. Dupret, "Event-based, timescale invariant unsupervised online deep learning with STDP," *Front. Comput. Neurosci.*, vol. 12, pp. 1–13, June 2018, doi: 10.3389/fncom.2018.00046.
- [53] R. Kreiser, D. Aathmani, N. Qiao, G. Indiveri, and Y. Sandamirskaya, "Organizing sequential memory in a neuromorphic device using dynamic neural fields," *Front. Neuromorphic Eng.*, vol. 12, pp. 1–17, Nov. 2018, doi: 10.3389/fnins.2018.00717.
- [54] R. Kreiser, P. Pienroj, A. Renner, and Y. Sandamirskaya, "Pose estimation and map formation with spiking neural networks: Towards neuromorphic SLAM," in *Proc. 2018 IEEE/Robotics Society Japan Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 2159–2166, doi: 10.1109/IROS.2018.8594228.
- [55] S. Glatz, R. Kreiser, J. N. P. Martel, N. Qiao, and Y. Sandamirskaya, "Adaptive motor control and learning in a spiking neural network, fully realized on a mixed-signal analog/digital neuromorphic processor," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA) 2019*. [Online]. Available: <https://arxiv.org/abs/1810.10801>
- [56] V. Braitenberg, *Vehicles: Experiments in Synthetic Psychology*. Cambridge, MA: MIT Press, 1984.
- [57] M. B. Milde, A. Diettmüller, H. Blum, G. Indiveri, and Y. Sandamirskaya, "Obstacle avoidance and target acquisition in mobile robots equipped with neuromorphic sensory-processing systems," in *Proc. 2017 IEEE Int. Symp. Circuits and Systems, (ISCAS)*, pp. 1–4, doi: 10.1109/ISCAS.2017.8050984.
- [58] M. B. Milde, D. Neil, A. Aimar, T. Delbruck, and G. Indiveri, "ADaPTION: Toolbox and benchmark for training convolutional neural networks with reduced numerical precision weights and activation," unpublished.
- [59] H. Blum, A. Diettmüller, M. Milde, J. Conradt, G. Indiveri, and Y. Sandamirskaya, "A neuromorphic controller for a robotic vehicle equipped with a dynamic vision sensor," in *Proc. Robotics: Science and Systems (RSS 2017)*, pp. 1–9, doi: 10.15607/RSS.2017.XIII.035.
- [60] E. Neftci, J. Binas, E. Chicca, G. Indiveri, and R. Douglas, "Systematic construction of finite state automata using VLSI spiking neurons," in *Biomimetic and Biohybrid Systems*, T. Prescott, N. Lepora, A. Mura, and P. Verschure, Eds. Berlin: Springer-Verlag, 2012, pp. 382–383.
- [61] Y. Sandamirskaya and G. Schöner, "An embodied account of serial order: How instabilities drive sequence generation," *Neural Netw.*, vol. 23, no. 10, pp. 1164–1179, Dec. 2010, doi: 10.1016/j.neunet.2010.07.012.
- [62] M. Luciw, S. Kazerounian, K. Lakhmann, M. Richter, and Y. Sandamirskaya, "Learning the condition of satisfaction of an elementary behavior in dynamic field theory," *Paladyn, J. Behavioral Robotics*, vol. 6, no. 1, pp. 180–190, Jan. 2015, doi: 10.1515/pjbr-2015-0011.
- [63] B. Duran and Y. Sandamirskaya, "Learning temporal intervals in neural dynamics," *IEEE Trans. Cogn. Develop. Syst.*, vol. 10, no. 2, pp. 359–372, June 2018, doi: 10.1109/TCDS.2017.2676839.
- [64] R. Kreiser, T. Moraitis, Y. Sandamirskaya, and G. Indiveri, "On-chip unsupervised learning in winner-take-all networks of spiking neurons," in *Proc. 2017 IEEE Biomedical Circuits and Systems Conf., (BioCAS)*, pp. 424–427, doi: 10.1109/BIOCAS.2017.8325168.
- [65] R. Kreiser, M. Cartiglia, J. N. Martel, J. Conradt, and Y. Sandamirskaya, "A neuromorphic approach to path integration: A head direction spiking neural network with vision-driven reset," in *Proc. 2018 IEEE Int. Symp. Circuits and Systems, (ISCAS)*, pp. 1–5, doi: 10.1109/ISCAS.2018.8351509.
- [66] C. Strub, G. Schöner, F. Wörgötter, and Y. Sandamirskaya, "Dynamic neural fields with intrinsic plasticity," *Front. Comput. Neurosci.*, vol. 11, pp. 74–87, Aug. 2017, doi: 10.3389/fncom.2017.00074.
- [67] D. Lobato, Y. Sandamirskaya, M. Richter, and G. Schöner, "Parsing of action sequences: A neural dynamics approach," *Paladyn, J. Behavioral Robotics*, vol. 6, pp. 119–135, May 2015, doi: 10.1515/pjbr-2015-0008.
- [68] Y. Sandamirskaya and T. Storck, "Learning to look and looking to remember: A neural-dynamic embodied model for generation of saccadic gaze shifts and memory formation," in *Artificial Neural Networks*, P. Koprinkova-Hristova, V. Mladenov, and N. K. Kasabov, Eds. Berlin: Springer, 2015, pp. 175–200.
- [69] S. Kazerounian, M. Luciw, M. Richter, and Y. Sandamirskaya, "Autonomous reinforcement of behavioral sequences in neural dynamics," in *Proc. 2013 Int. Joint Conf. Neural Networks (IJCNN)*, pp. 1–8, doi: 10.1109/IJCNN.2013.6706877.
- [70] C. Eliasmith and C. Anderson, *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*. Cambridge MA: MIT Press, 2004.
- [71] M. Payvand, M. Nair, L. Müller, and G. Indiveri, "A neuromorphic systems approach to in-memory computing with non-ideal memristive devices: From mitigation to exploitation," *Faraday Discussions*, vol. 213, pp. 487–510, July 2018, doi: 10.1039/C8FD00114F.
- [72] G. Indiveri and S.-C. Liu, "Memory and information processing in neuromorphic systems," *Proc. IEEE*, vol. 103, no. 8, pp. 1379–1397, Aug. 2015, doi: 10.1109/JPROC.2015.2444094.
- [73] C. Liu, G. Bellec, B. Vogginger, D. Kappel, J. Partzsch, F. Neumärker, S. Hoppner, W. Maass et al., "Memory-efficient deep learning on a spinnaker 2 prototype," *Front. Neurosci.*, vol. 12, pp. 840–855, Nov. 2018, doi: 10.3389/fnins.2018.00840.
- [74] M. B. Milde, H. Blum, A. Diettmüller, D. Sumislawska, J. Conradt, G. Indiveri, and Y. Sandamirskaya, "Obstacle avoidance and target acquisition for robot navigation using a mixed signal analog/digital neuromorphic processing system," *Front. Neurobotics*, vol. 11, no. 28, 2017, doi: 10.3389/fnbot.2017.00028.

Shih-Chii Liu, Bodo Rueckauer, Enea Ceolini,
Adrian Huber, and Tobi Delbruck

Event-Driven Sensing for Efficient Perception

Vision and audition algorithms



©ISTOCKPHOTO.COM/JUST_SUPER

Event sensors implement circuits that capture partial functionality of biological sensors, such as the retina and cochlea. As with their biological counterparts, event sensors are drivers of their own output. That is, they produce dynamically sampled binary events to dynamically changing stimuli. Algorithms and networks that process this form of output representation are still in their infancy, but they show strong promise. This article illustrates the unique form of the data produced by the sensors and demonstrates how the properties of these sensor outputs make them useful for power-efficient, low-latency systems working in real time.

Event sensors

With increasing access by many groups to event-driven spiking sensors, such as silicon retina event cameras [1] and silicon cochlea audio sensors [2], interesting event-driven vision and auditory algorithms are being developed to deal with this new type of dynamically sampled output. Some of these algorithms, including machine-learning deep networks, have led to systems that successfully use these continuous-time asynchronously sampled inputs on tasks, such as quick auditory localization [3] and quick visually servoed robots [4]. Event sensors abstract the function of biological sensors, such as eyes and ears. These organs locally sense a signal (light or sound), convert it to an asynchronous digital neural spike representation, and send these spikes on to the central nervous system. We call the electronic analog of biological spikes *events* to indicate that they are asynchronous in time and that they carry along information about the location and time.

Useful event sensors generally produce quite sparse (i.e., nonredundant) output. For example, the Dynamic Vision Sensor (DVS) event camera produces events when pixels detect a significant change in log intensity. As a result, the sensor produces a sparse asynchronous output in a typical natural scene, leading to immediate savings in both latency and the amount of computation needed by the postprocessing system.

Spiking neural networks from computational neuroscience are natural postprocessor candidates of the sensor events. However, recent achievements on various vision and audition tasks

are obtained by adapting conventional machine-learning algorithms such that the advantages of the low-latency and sparse properties of the asynchronous outputs from the event sensors are combined with the widespread availability of supervised deep-learning training methods that quickly achieve accurate networks. The sparse output of event sensors is ideally matched to upcoming hardware accelerators for conventional deep neural networks that exploit activation sparsity.

Advantages of event sensors

Event sensors, unlike conventional regular-sampled sensors, act as the driver of their output data. As with their biological counterparts, the pixel elements of the sensors autonomously transmit events that carry information about the external world. This form of encoding has three advantages. First, the self-sampling allows wide dynamic range because the integration time is set by the pixel rather than by a fixed sample rate. Second, properly designed sensors transmit only informative events, which reduces both signal redundancy and system power in the postprocessing system. Third, these events are transmitted with low latency because the communication is initiated by sensor pixels. That allows event sensor systems to beat the latency versus power tradeoff of sampled sensor systems, as detailed in the “Latency and Power Tradeoffs for Event Sensors” section. Besides vision and audio sensors, event sensors for other modalities, such as chemical [5] and tactile [6] sensors, have been investigated as well. Here, we briefly outline the properties of the event cameras and audio event sensors.

Event cameras (silicon retinas)

The DVS, which is the most developed event sensor, has several offshoots, such as the higher sensitivity DVS, and cameras that produce both an event-driven output and an intensity output [1], [7]. Figure 1 shows a variant of the latter, the Dynamic and Active Pixel Vision Sensor (DAVIS) camera with two outputs. The first output consists of asynchronous streams of temporal-contrast brightness change events seen by the pixels. On and Off events encode positive and negative

log intensity changes that exceed a defined threshold following the known send-on-delta scheme. The second output is the intensity readout as part of a regularly sampled frame output of a typical camera.

Audio event sensors (silicon cochleas)

Another highly developed event sensor is the Dynamic Audio Sensor (DAS), a silicon cochlea that produces events from a set of bandpass filters modeling the biophysics of the basilar membrane along with circuits that model the properties of the inner hair cells and produce the spiking output of the auditory nerve fibers of the biological cochlea [7].

Data-driven dynamic sampling

Conventional cameras read out an entire frame of pixels that are uniformly sampled using a global clock. This type of sampling is the standard paradigm in the design of analog-to-digital converter (ADC) systems. Its implementation with clocks is efficient, and the mathematical basis necessary for properly analyzing the sampled signals is well understood. The frame rate in conventional cameras has to be large enough such that fast motions at any point in the visual field of the camera can be resolved. Natural scenes, however, contain motions of varying and different velocities, and choosing a frame rate matched to the fastest observed motion is inefficient. A sampling scheme where each pixel sets its effective sampling rate independent of other pixels can be more efficient, insofar as the local pixel sampling rate is matched to the corresponding motion in visual space. Data-driven dynamic sampling can, therefore, directly exploit the dynamic nature of visual scenes. An example of this type of data is discussed in the “Data Compression” section.

Once uniform sampling is given up as a consequence of data-driven sampling schemes, difficulties emerge in how to process the now asynchronously sampled signal optimally. The conventional assumption that justifies sampling is that the sampled signals are bandlimited. Under this assumption, it can be shown that to obtain a stable mapping from the bandlimited function to its sample values, the sampling rate must in any case

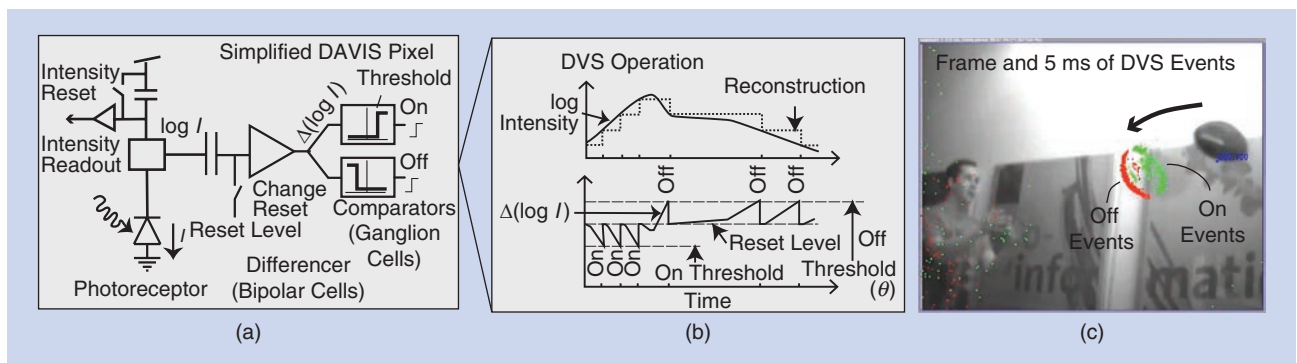


FIGURE 1. The DAVIS event camera. (a) The pixel photoreceptor converts a photocurrent I into a log intensity voltage $\log(I)$ while also integrating I over time for intensity readout. The On and Off DVS brightness change events are generated by the differencer and comparator circuits that model the bipolar and ganglion cells of biological retinas. (b) A continuous-time log intensity signal is converted to a sequence of On and Off events, $e(t)$. The event stream can be used for a simple step reconstruction of the original log intensity. (c) The two output streams (events and frames) for a natural scene with a football thrown to a person. The gray-level frame is shown together with the brightness change events (red and green dots), which are output during the 5 ms while the next frame is being exposed.

be, on average, larger than the Nyquist–Landau rate (both for uniform as well as nonuniform sampling) [8]. Therefore, sampling methods can be differentiated in terms of the energy efficiency of the complete system implementation but not in terms of the number of samples that are generated if a mathematically proper representation of the sampled signal is desired.

The assumption of bandlimitedness, however, is fulfilled by natural signals to different degrees. Audio signals, for example, are canonical examples of bandlimited functions. Vision signals, on the other hand, often lack this structure: a white rectangle on a uniform black background is not bandlimited in space, and if the rectangle moves, it is also not bandlimited in time. A physical camera, in contrast, has an extended point spread function, smoothing such sharp edges and, thereby, approximately band-limiting the visual signal. But the assumption of bandlimitedness is still less justified than for audio signals, such as speech, which are inherently bandlimited irrespective of the recording device.

Thus, vision can profit directly from sampling schemes that are nonuniform and matched to the particular structure of the recorded object. For example, for the white rectangle on a uniform black background, it is sufficient to record only the edges. A sampling scheme that does this captures the scene more efficiently than a uniform clocked sampler. In contrast, because audio signals are more bandlimited than vision signals, it is more difficult to find a sampling scheme that is more efficient than uniform sampling on the level of signal representation.

In addition to the added value of locally adaptive sampling rates, dynamic sampling schemes can be tailored to the nature of the observed signal more efficiently than clocked samplers. One example of a dynamic data-driven sampling scheme is the send-on-delta sampling scheme [9] that is used in the DVS event camera. In this sampling scheme, sampling values are taken only if the sampled signal has changed sufficiently since the last measured sample value. Therefore, this sampling method leads to a large number of samples in regions in which the signal changes rapidly and to few samples in the opposite case (see Figure 1). When send-on-delta sampling is applied to bandlimited signals, the resulting set of sampling times (and the corresponding implicitly obtainable sampling values at these times) are not sufficient to reconstruct the original function from the samples. This statement can be proved rigorously [10], but send-on-delta sampling can provide efficiency advantages if precise reconstruction is not necessary.

To derive signal processing statements, it is implicitly assumed that both uniform and nonuniform sampling methods output the amplitude values at the sampling points. In the send-on-delta sampling scheme, however, only the time at which a sample is recorded is kept. However, due to the sampling mechanism amplitude information is retained and can be extracted from the temporal information. Therefore, for send-on-delta sampling, the spike information is equivalent to amplitude information, justifying an analysis of the sampling scheme with the language of classical signal processing.

Implementations of asynchronous sampling systems

Data-driven dynamic sampling as a paradigm can be implemented by various means. In essence, the analog output

of a local pixel of a physical sensor (e.g., a camera pixel, a channel of a multifrequency filter cochlea) can be made event driven by converting the local continuous-time output through an asynchronous ADC circuit and transmitting these events close to the time that they occur. These architectures contrast with uniform sampling, in which a regular clock drive reads out the intensity value of the pixels of a camera in a row-like format and the values are digitized by synchronous ADCs at the periphery of the array. Asynchronous sigma-delta converters are well established, but, because of their circuit size, other asynchronous schemes, such as send on delta, level crossing, and integrate-and-fire neuron models [11], are used more often within the pixel.

Data compression

Figure 2 shows an example of how using the DVS leads to reduced recording data from an overhead view of a mouse in a cage. A researcher might be interested in studying the whisker motions of the mouse. These motions require a high sample rate of 1 kHz, which is easily supplied by the DVS pixel bandwidth. We observe from the 3.5-day recording that there are long periods of near silence from the sensor followed by brief bursts of high activity as the mouse explores the cage. The event rate probability follows a distribution where the log probability decreases approximately linearly and inversely with the event rate. This means that the output event rate is dominated by mostly low rates, with a tail of high rates encoding the rapid motions. A 1-kfps image sensor would produce about 300 Mframe or 5 terabytes of data from the equivalent resolution 128×128 pixel sensor. The DVS recorded 18.5 million events, or 74 megabytes of 4-byte event data, which is a factor of 67,000 times less data. Of course, a standard camera output could be greatly compressed, but its output would require at

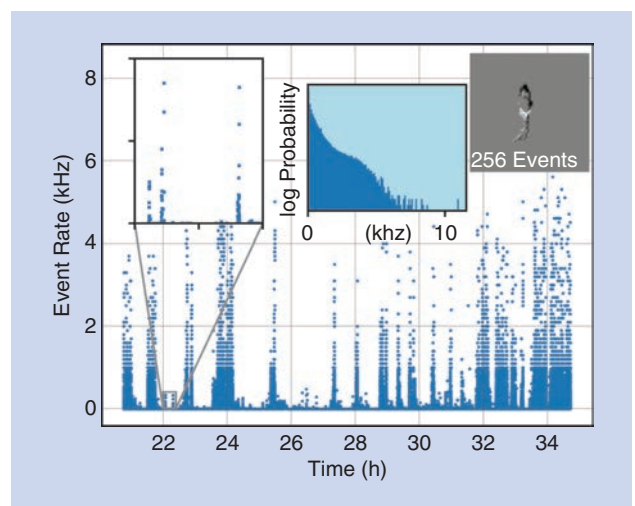


FIGURE 2. The event rate of a DVS recording of the overhead view of the mouse (right inset) over 14 h. Peaks correspond to high levels of motion and, therefore, potential regions of interest. The middle inset shows a histogram of the event rates for the complete 3.5-day recording, using a log scale for the vertical axis. (Data from unpublished collaboration with Irene Tobler, University of Zürich.)

least initial processing at this high frame rate, and this processing (and its latency) is avoided by using a DVS camera.

Latency and power tradeoffs for event sensors

Limiting system response latency is essential in some applications. In vision, each frame contains many pixels and can be highly informative. Response latency is limited by frame interval. In audio, inference typically depends on many samples or, equivalently, many audio frames, which are often short-time spectrograms. Therefore, although low-latency audio processing is sometimes advantageous, this low latency typically is a result of postprocessing algorithm development rather than being determined by an audio sample or a frame interval.

When latency depends on a sample interval (as in vision), reducing latency L requires sampling and processing at a higher rate (i.e., running at higher frame rate R). Therefore, there is a fundamental latency-power tradeoff: a lower latency requires higher power. We will characterize this tradeoff quantitatively using the power-latency product (PLP). The PLP has units [energy/time] \times time = energy, and the more efficiently the system performs the task at a particular average latency, the smaller the PLP. (In digital electronics, an analogous quantity called *power-delay-product* characterizes the efficiency of logic gates in achieving a delay.)

We now compare the PLP for a frame-based and event-based visual inference problem. Using frames, we assume that each frame provides sufficient information to infer a response and that the camera must run at a frame rate R to achieve sufficiently short latency. With standard camera pipelines, frame processing can start only after the frame is captured and fully transferred. Then, the average latency using frames will be

$$L_{\text{frame}} = \frac{1}{2R} + T_{\text{process}}, \quad (1)$$

where T_{process} is the time required to process the frame. The fraction $1/2$ of the frame interval $1/R$ comes from the fact that on average, the relevant external event the system must react to occurs halfway between the frames. If each frame requires energy E_{frame} to process, then

$$\text{PLP}_{\text{frame}} = RE_{\text{frame}}L_{\text{frame}} = (1/2 + RT_{\text{process}})E_{\text{frame}}. \quad (2)$$

We can observe that $(1/2)E_{\text{frame}} < \text{PLP}_{\text{frame}} < (3/2)E_{\text{frame}}$. The minimum value of $\text{PLP}_{\text{frame}}$ occurs when processing takes zero time, and the maximum occurs when processing occupies the entire time between frames. Using events, we first assume that only a fraction F of total pixels that produce events are needed for inference. We further consider that the cost of processing these events is $E_{\text{event}} = FE_{\text{frame}}$ and that these events occur in a time that is $1/R$ at most (i.e., we assume that the events are generated quickly enough to at least achieve the desired minimum latency). Then the latency will be $1/R$, and the rate of processing these chunks of events will be R , shown as follows:

$$\text{PLP}_{\text{event}} = RFE_{\text{frame}}1/R = FE_{\text{frame}}. \quad (3)$$

Here we infer that the processing occurs concurrently with the capture of the events so that there is no extra T_{process} . This concurrency is practical on existing standard hardware by using

overlapping USB data transfer. In existing USB event cameras, packets of events are transferred at a maximum interval of 1 ms, and the packet rate can go up to at least 8/ms, enabling submillisecond response latencies. This analysis makes it clear that using events results in a significantly lower PLP for problems with sufficiently sparse F . For example, if $T_{\text{process}} = 1/(2R)$ (i.e., processing the frame takes half the frame interval) and $F = 0.1$, then $\text{PLP}_{\text{frame}}/\text{PLP}_{\text{event}} = 10$, so using events provides a $10\times$ improvement in PLP. The gain in PLP comes from sparsity and the concurrent processing of events.

Feature extraction from events

Feature extraction is a common preprocessing step in many sensor processing tasks. Although it is certainly possible to directly operate on the level of individual events, a separate preprocessing step can help to improve the efficiency of the entire algorithmic pipeline. For example, reducing DVS noise by filtering out events using spatiotemporal correlation reduces the number of uninformative events [12] and can require only 1D memory at the edge of a 2D pixel array [13].

Model-based feature extraction

One popular approach to extract meaningful features is to assume knowledge of how the spike event was generated or a model of how events should be aggregated together to form a representation. These models typically make use of the high temporal precision offered by the outputs of the event-driven sensors. For example, orientation filters can be based on the interpixel and intrapixel timing of spikes within a spatial neighborhood [12], with later work showing the use of this timing information for stereo matching. Models based on computer vision formulations use a data structure variously known as a *time-stamp image*, *time surface*, or *surface of active events* to supply a spatiotemporal 2D array on which to operate. These structures are applied to compute optical flow and a 3D pose, for example, as in [14].

Recent feature extraction methods for these event sensors include the use of Bayesian formulations in tractable forms that allow real-time implementation. These methods model the generation of DVS events as probabilistic processes that depend on camera movement, spatial contrast in the scene, and noise. The state (camera pose, spatial contrast, and 3D structure) can be updated from each event in a principled manner. This approach was successfully applied on vision tasks, such as scene reconstruction [15], [16] and camera pose estimation [17]. The event timings from the output of a binaural DAS have also been used successfully to support a Bayesian model of sound localization [3].

Supervised and unsupervised feature learning

Unsupervised feature extraction methods where the model parameters are adjusted even in the absence of target labels have been introduced. A subset of these algorithms use local unsupervised spike-based learning rules from computational neuroscience to transform DVS data into useful representations for a task [18]. Other methods include the clustering of events into spatiotemporal time surfaces [19] and applying them to high-speed tracking as well as object recognition tasks.

Supervised learning methods are also used to obtain features that implicitly arise during learning. Methods based on computational neuroscience concepts include ways of projecting the input events to a higher-dimensional space using a set of time-varying temporal synaptic kernels and then optimizing a linear combination of these projections for the output neuron values based on the target labels [20]. A family of algorithms frequently used for such implicit feature extraction are deep neural networks (see the section “Deep-Learning (Algorithms) With Event Sensor Input”).

Feature event frames

Whatever features are used, collecting them into frames is useful because it increases efficiency and brings a set of distinct events into context with each other. Frames of event sensor data (event frames) can be created from asynchronous events by accumulating a variable number of events over a fixed time window (constant time frames) or over a fixed number of events, resulting in a variable time window (constant count frames). Assume that the event stream is described as $e_i = [t_i, f_i]$, $i \in \mathbb{N}$, where e_i is the i th event from the channel f_i in the event stream at time t_i . For constant time frames F^{tf} the j th time frame of length T_l is defined as

$$F_j^{tf}(f) = \text{card}(\{e_i | T_l \cdot (j-1) \leq t_i < T_l \cdot j, f_i = f\}), \quad (4)$$

where $\text{card}()$ is the cardinality of a set, and f is the channel number. For constant count frames F^{ef} of E events each, the j th frame is defined as follows:

$$F_j^{ef}(f) = \text{card}(\{e_i | E \cdot (j-1) \leq i < E \cdot j, f_i = f\}). \quad (5)$$

Other strategies that preserve appropriate features include, for example, a time-stamp image, $\tau_i(t, f)$, which is defined as follows:

$$\tau_i(t, f) = \max(t_i, f_i = f) \quad (6)$$

for $t_i < t$. Time-stamp image frames can help to create orientation filters [12] or be used in inferring the optical flow between two conventional vision frames [see the “Deep-Learning (Algorithms) With Event Sensor Input” section].

The type of frame to use depends on the task. Constant count frames of DVS events, for example, reduce motion blur compared with constant time frames, because the faster the motion, the briefer the frame. Using constant count frames results in a frame sampling rate that is driven by the input dynamics and gives better classification accuracies than constant time frames on a particular object tracking task using the DVS [21]. Constant count frames do not give better accuracy on all tasks. For example, on an audio digit recognition task, using either constant time frames or constant count frame features on DAS events did not lead to a significant difference in the classification accuracy [22].

Tradeoff between accuracy and computational cost

To better understand the accuracy and computational cost tradeoff using event sensors, we carried out two experiments that are discussed next in the “Vision Task” and “Audio Task” sections.

Vision task

We look at the accuracy and cost tradeoff when computing optical flow directly from the events and from frames based on events. For this experiment, we use constant time frames so that we can choose arbitrary frame rates. The experimental setup is as follows: a white disk with a black bar near its outer edge is mounted on a motor with controllable speed. A DAVIS camera records the disk while we ramp the rotational speed up and down. The rotational frequency of the disk increases to a maximum of about 12 Hz within 3 s, holds that level for another 5 s, and finally decreases continuously to 5 Hz over a period of 7 s. The leading edge of the dot on the disk generates a stream of Off events from the DVS. We filter out the On events from the trailing edge, which are not needed for this task. Background events are also discarded using a spatiotemporal correlation criterion [12]. The median event rate is 120 kHz. Figure 3(a) illustrates the experimental data.

For event-based flow computation, we used the Savitzky–Golay variant of the LocalPlanes algorithm described in [23]. The algorithm maintains a time surface corresponding to the most recent event time stamps for all pixels (see the section “Model-Based Feature Extraction”) and attempts to fit a 2D plane in the 3×3 neighborhood around each incoming event pixel address. The normal flow of the edge causing the event is then obtained from the slope of the fitted plane.

We estimate the rotational frequency from the flow vectors by projecting them onto the tangent space of the circular trajectory. The event data curve in Figure 3(b) shows the resulting rotation frequency over the duration of the recording. Of course, knowing that we are observing a rotating dot, it is clear that the rotation frequency of the disk could have been obtained from methods other than flow, but we use this metric as a proxy for the accuracy of the flow computation.

For the frame-based flow computation, we use the standard OpenCV implementation, which consists of a key-point detector combined with a Lucas–Kanade (LK) sparse flow algorithm. Parameters of the LK algorithms are optimized to obtain the highest accuracy. We generated frames for the LK algorithm from the continuous DVS event stream at frame rates varying from 50 to 300 Hz. Each frame is a 2D histogram of the DVS events collected during the frame interval (refer to the “Feature Event Frames” section).

Figure 3(b) shows that because the event stream is near continuous (with a time resolution of $1 \mu\text{s}$), the event-based flow algorithm does not suffer from aliasing and accurately estimates the motion at all speeds (blue squares). The frame-based flow algorithm is able to accurately estimate the optical flow over the whole duration of the recording, but only if the frame rate is 100 Hz or more. At the lower frame rate of 50 Hz, fast motion (greater than 10 cycles/s) results in large displacements of the rotating bar across consecutive frames and, consequently, high errors of the LK algorithm. This tradeoff between frame rate (a proxy for computational cost) and accuracy is demonstrated in Figure 4, which shows how the mean-square error (MSE) of the estimated rotational frequency decreases with higher sampling rates.

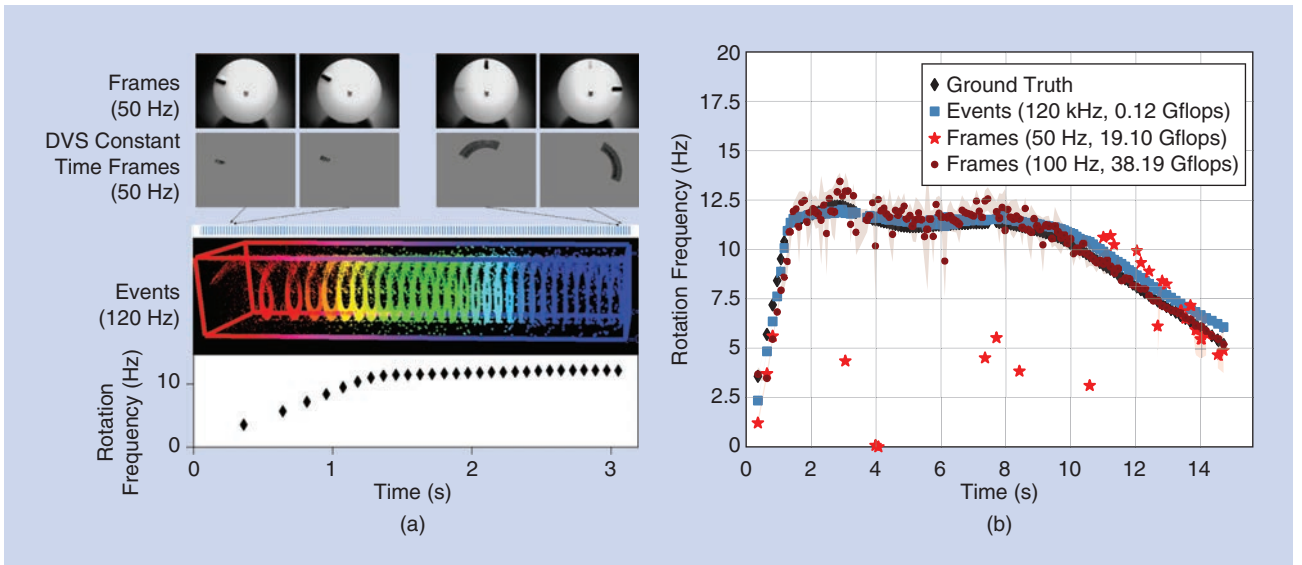


FIGURE 3. (a) The illustration of the spinning disk data. (b) The rotational frequency measurements from optical flow over the duration of a DVS recording of a spinning disk, estimated using an event-based optical flow algorithm (blue) and a frame-based LK method (red). The ground truth (black diamonds) for the motion of the disk is obtained by measuring the time stamps of events produced when the dash passes a 1×30 -pixel window and then k -means clustering the times of events obtained in this window. The time stamps of a single pixel initialize the k -means algorithm.

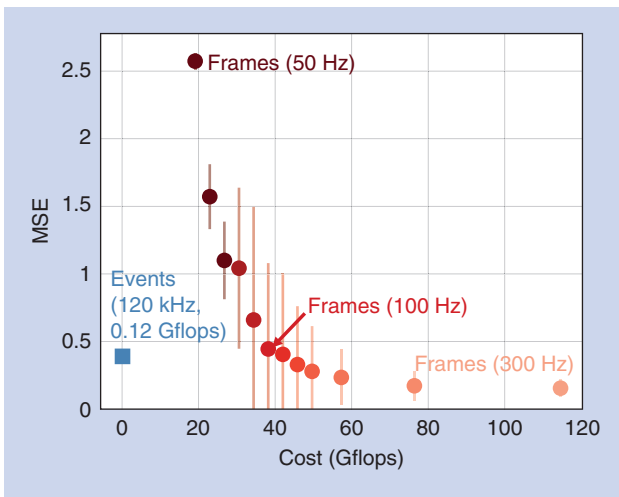


FIGURE 4. The MSE of the extracted rotational frequency against the computational cost with different event frame rates (circular disks color coded in red). The blue square marks the accuracy and cost of the event-based flow algorithm.

The frame-based sparse LK algorithm requires 382 Mflop per frame, while the event-based flow computation costs 980 flop per event [23] and, therefore, an average of 0.12 Gflops for this recording. To achieve the same level of accuracy as the event-based method, the frame-based algorithm requires a frame rate of at least 100 Hz, costing about 38 Gflops, which is a factor of approximately 320 times more. This experiment illustrates the advantage of data-driven dynamic sampling, which, in this case, enables the event-based flow algorithm to accurately solve the task at high object velocities without special knowledge of the rotating dot model that could accommodate large displacements for fast motion.

Audio task

For an audio task, we study a similar accuracy-latency tradeoff as for the video task. For this case, we analyze the tradeoff among accuracy, cost, and time-to-first classification in a localization task using the timing of the audio arriving at two microphones. We compare the performance of a Bayesian localization model that uses silicon cochlea DAS events [3] (referred as *event sampling*) against a classical localization approach, namely, generalized cross-correlation (GCC) with phase transform that uses classical uniform sampling. For the task, we used a subset of the data set recorded in [3]. Specifically, a loudspeaker plays audio book samples at a signal-to-noise ratio level of -5 dB in the presence of babble noise played from the other loudspeakers. The position of the loudspeaker varies among seven different angles in the set (0° , 30° , 60° , 90° , 120° , 150° , and 180°) arranged in a semicircle of a 2-m radius. At the origin of the circle, we placed the DAS microphones separated by 11 cm and two microphones separated by 15 cm uniformly sampled at 44.1 kHz.

To compute the localization output using events, we use the difference in arrival times between the events from the two microphones to compute an interaural time difference (ITD) estimate. These ITD events are then used to update the Bayesian localization model of the speaker positions. For the uniform samples, we compute the localization output using GCC. We first calculate the cross-correlation over a certain frame size and map the delays onto the seven possible delays corresponding to the seven loudspeaker positions. The final localization output corresponds to the position with the highest correlation.

Figure 5 displays the results for the localization task using the two approaches. In particular, we show the accuracy in localization for different frame sizes, emphasizing the accuracy after the time-of-first classification for each of the conditions. The plot also shows the change in localization accuracy when evidence

is accumulated over 1 s. With increasing frame size, the time-to-first classification is delayed because one needs to wait for at least the amount of time needed to collect a full frame before calculating the first classification output. From then on, we integrate the evidence collected by moving frames with a shift of 8 ms.

The results show that for short latencies (<500 ms), the DAS localization model is more accurate than GCC. It is known that GCC does not perform very well when using short frames [24]. However, if one is willing to wait longer, GCC accuracy is slightly higher. But if one can afford only a short latency, the DAS provides better accuracy. When quick responses to changes in the acoustic scene are necessary, then event sampling can provide an advantage.

The cost of time-to-first classification can be defined as the number of flop needed to obtain the classification value. The DAS localization algorithm requires a number of operations that are linear in the number of events, while the GCC requires a number of operations that are $O(n \log n)$ due to the fast Fourier transform calculation. For this example, the processing of event samples is at least five times less expensive than the processing of uniform sampling via GCC for an equivalent sampling rate.

Deep-learning (algorithms) with event sensor input

As briefly mentioned in the “Feature Extraction From Events” section, various algorithms have been developed for the post-processing of the asynchronous sensor events, ranging from neuroscience-based spiking neural networks to current developments exploiting advances in deep learning. Deep learning has surpassed traditional machine-learning methods on many vision and audio benchmarks [25], capitalizing on the success of training algorithms and massive labeled data sets. Here, we focus mostly on deep network algorithms because these methods have proven to be useful for systems that can be deployed in complex real-world scenarios.

There are several examples of how event sensors driving deep networks result in systems that are quick and of low power. For example, we used an event camera to drive a convolutional neural network (CNN) to allow a predator robot to follow a prey robot [21]. The DVS frames drive the inference at a variable rate from 1 to 240 Hz depending on the speed of the robot. Another example is a hand symbol recognition robot that plays the game of rock, paper, scissors. It shows a real-time application of a DVS driving a field-programmable gate array hardware implementation of a five-layered CNN [26]. The CNN detects four classes (rock, paper, scissors, and background). With fewer than 10 ms of latency, the DVS CNN creates the illusion that the machine always outguesses the human. The use of constant count frames [see the examples in Figure 6(a)] on this robot prevents blurring of the hand during fast motion while also ensur-

ing that the DVS frames have useful information and maximizing the laptop battery lifetime. Recurrent neural networks are usually used more often on temporal sequence tasks, such as automatic speech recognition. An example of applying them to DAS output is in the simple digit recognition demonstration of [22].

Spiking neural networks (SNNs), which form the substrate of many neuroscience models, can benefit from the advances made in using analog neural networks (ANNs) that have been trained on event sensor outputs. The approximation of an ANN unit with an SNN unit is shown in Figure 6(b). The development of methods to approximate pretrained ANNs by SNNs has led to [27]–[29], which demonstrate that commonly used ANNs can be converted into spiking networks with minimal accuracy loss, including the widely used VGG-16, ResNet, and GoogLeNet Inception-v3 nets, although with increased computation due to the discretization of hidden layer activations into binary spike events. A thorough review of the SNN training and conversion literature is given in [30].

Labeled data sets collected from the event sensors are usually small compared to well-established deep-learning data sets. However, deep-learning algorithms can be applied as a self-supervised method for training networks that indirectly extract information, such as optical flow, in challenging scenes where the timing from the DVS is useful. From both frames and the asynchronous DVS events generated from the DAVIS camera, algorithms such as the EV-FlowNet [31] indirectly give the predicted flow from one normal frame to the next by using the time-stamp images derived from the events.

Discussion

This article presents case studies comparing the processing of event-based and standard frame-based sensor output. Event sensors are not optimal for all applications. Table 1 lists a set of advantages and disadvantages between frame-based cameras and the DVS. When power, latency, and dynamic range are not important, then a frame-based approach is preferable.

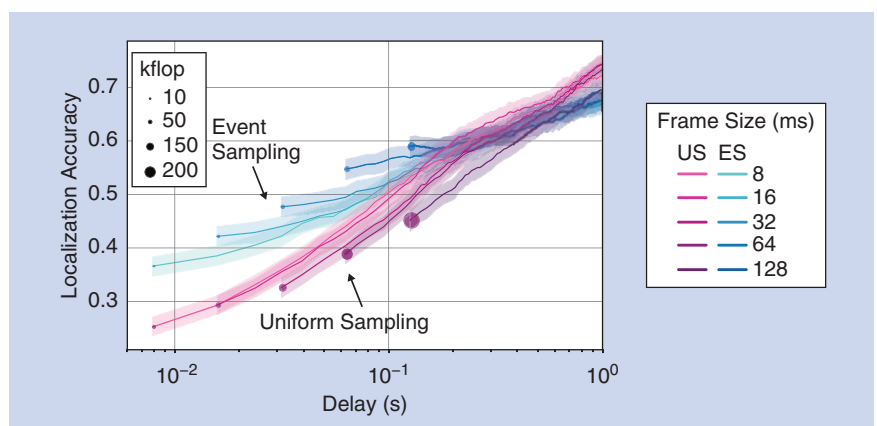


FIGURE 5. A comparison of accuracy and latency using uniform and event sampling methods on the audio localization task. The plot shows localization accuracy versus integration time (latency) with different frame sizes. In all cases, a frame shift of 8 ms is applied. Because of the frame size, the time-to-first classification is different for each frame size. Solid circles indicate the time-of-first classification, and their size indicates the computational cost in kflop. Shading shows the standard deviation over the data set. ES: event sampling; US: uniform sampling.

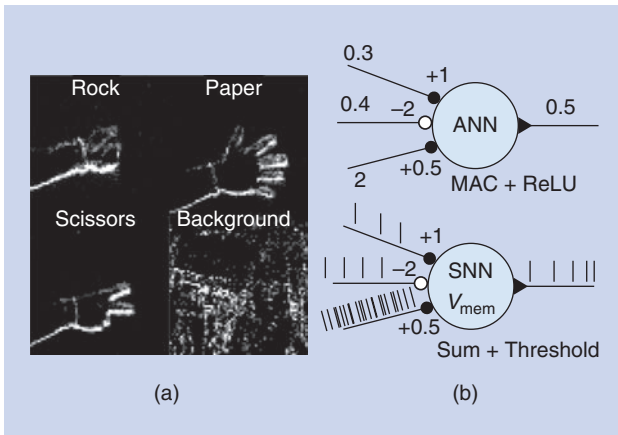


FIGURE 6. (a) The examples of the four classes of the rock, paper, scissors game, using constant count DVS frames. (b) An approximation of an ANN with an SNN unit. The input analog value for each connection is approximated with the corresponding number of events in a chosen time window. The weights of each connection are listed next to the small dark or white circle to the blue ANN or SNN circle. The inputs are weighted and summed up using multiply-accumulate (MAC) operations and then passed through a rectified linear unit (ReLU) producing the ANN output. For the SNN unit, an input event is weighted by the value of that connection. An output event is produced whenever the weighted sum of the events exceeds a threshold.

New developments of asynchronous event-driven signal processing methods would be complementary to learning-based approaches [see the “Deep-Learning (Algorithms) With Event Sensor Input” section] of processing event data and would, for example, allow one to define notions, such as filtering of event data. Operations, such as state estimation via Bayesian filters (e.g., Kalman filters), can naturally deal with irregular sampling patterns from event sensors as long as the dynamics of the system generating the observables are known [32]. As such, Kalman filters can be applied to the output of event sensors if a known dynamical system is observed.

The development of new signal processing methods is necessary because existing signal processing methods, such as filtering, are applied primarily to uniformly sampled signals, where continuous convolution is mapped onto discrete-time convolution. It is important that some signal structure is preserved when moving from continuous time to discrete

time so that convolution values are sufficiently close to continuous-time convolution. It is an open question how to design filters, especially ones that do not require resampling and that achieve the equivalent of a continuous-time filter when they are applied to the output of the event sensor. See [33] for a comparison of event-driven versus frame-based spatial filtering for an event camera. For filter design to be possible, it is necessary to understand for which classes of signals event-based sensors are information preserving (i.e., the class of signals for which, after sampling, sufficient information is retained in the samples to have a sufficiently accurate convolution).

Another major driver of the advancement of event-driven deep networks is the increasing number of data sets from the event sensors (e.g., standard vision data sets that are converted to spiking data sets by playing the images to the DVS, such as the N-MNIST, and video-based data sets, such as the 2015 Visual Object Tracking challenge and UCF50 action recognition, which are more suitable to the DVS sensor). Examples of real-world event sensor data sets include gesture, driving, pose estimation, and rock, paper, scissors [4]. These data sets have been used to drive the development of algorithms for event sensors (e.g., optical flow computation as well as robot navigation and localization).

Besides the processing of event-based data with deep-learning methods, many practical tasks can be solved by more conventional methods (e.g., Bayesian filters) [3], [14]–[17]. Such methods are advantageous if a precise understanding of the algorithm used is required and the physics of the measured object are known. Precise understanding of the more-opaque deep-learning approaches is not always possible. Advancements in signal processing algorithms are, therefore, necessary to handle this new type of dynamically sampled data. Such advancements will allow more effective processing of the outputs of event-driven power-efficient sensors.

Acknowledgments

We thank the members of the Sensors group (Daniel Neil and Jithendar Anumula in particular), the University of Zürich, the Samsung Institute of Advanced Technology, and the Swiss National Science Foundation.

Table 1. Advantages and disadvantages of the DVS.

| Frame-Based Camera | DVS | |
|----------------------------------------------------------|---------------------------------------------------------------------------|----------------------------------------------------------------------------------|
| Attributes | Advantages | Disadvantages |
| Frames | Continuous event stream | Many events for textured scenes, high bandwidth needed when events come together |
| Highly redundant data | Direct low-bandwidth event output | Difficult reconstruction |
| Large suite of image processing algorithms | Potential for algorithms that capitalize on timing | Less availability of algorithms with guaranteed performance |
| Delay (minimum latency in one frame, a few milliseconds) | Instantaneous features (minimum latency of one event, a few microseconds) | — |
| Limited dynamic range (e.g., 60 dB) | High dynamic range (>120 dB) | Coarse quantization (e.g., each event represents a 10% change) |

Authors

Shih-Chii Liu (shih@ini.phys.ethz.ch) received her Ph.D. degree in computation and neural systems from the California Institute of Technology, Pasadena, in 1997. She worked at various companies, including Gould American Microsystems, LSI Logic, and Rockwell International Research Labs. Currently, she is a professor with the Institute of Neuroinformatics at the University of Zürich, Switzerland. She is a Senior Member of the IEEE.

Bodo Rueckauer (rbodo@ini.uzh.ch) received his B.Sc. and M.Sc. degrees in physics from ETH Zürich in 2014 and 2016, respectively. He is currently a Ph.D. candidate at the University of Zürich and ETH Zürich. His research interests include event-based processing with deep spiking neural networks as well as continual, local learning.

Enea Ceolini (eceoli@ini.uzh.ch) received his B.Sc. degree in electrical engineering and information technology from the University of Padova, Italy, in 2013 and his M.Sc. degree in neural system and computation from the University of Zürich and ETH Zürich in 2016. He is currently a Ph.D. candidate at the University of Zürich and ETH Zürich working on signal processing and machine learning for traditional and event-based audio processing. He is a Student Member of the IEEE.

Adrian Huber (huberad@ini.uzh.ch) received his B.Sc. and M.Sc. degrees in electrical engineering, information technology, and computer engineering at RWTH Aachen University, Germany, in 2013. He is currently a Ph.D. candidate at ETH Zürich and works on sampling theory and reconstruction.

Tobi Delbruck (tobi@ini.phys.ethz.ch) received his B.Sc. degree in physics from the University of California, San Diego, in 1986 and his Ph.D. degree from the California Institute of Technology, Pasadena, in 1993. He is currently a professor of physics and electrical engineering with the Institute of Neuroinformatics at ETH Zürich, where he has been since 1998. His group with Shih-Chii Liu focuses on neuromorphic sensory processing and efficient deep learning. He is a Fellow of the IEEE.

References

- [1] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck, "Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output," *Proc. IEEE*, vol. 102, no. 10, pp. 1470–1484, Oct. 2014. doi: 10.1109/JPROC.2014.2346153.
- [2] S.-C. Liu, A. van Schaik, B. A. Minch, and T. Delbruck, "Asynchronous binocular spatial audition sensor with $2 \times 64 \times 4$ channel output," *IEEE Trans. Biomed. Circuits Syst.*, vol. 8, no. 4, pp. 453–464, 2014.
- [3] J. Anumula, E. Ceolini, Z. He, A. Huber, and S.-C. Liu, "An event-driven probabilistic model of sound source localization using cochlea spikes," in *Proc. 2018 IEEE Int. Symp. Circuits and Systems*, pp. 1–5.
- [4] G. Gallego et al., "Event-based vision resources," GitHub, 2017. [Online]. Available: https://github.com/uzh-rpg/event-based_vision_resources
- [5] A. Vanarse, A. Osseiran, and A. Rassau, "An investigation into spike-based neuromorphic approaches for artificial olfactory systems," *Sensors*, vol. 17, no. 11, pp. 1–16, Nov. 2017. doi: 10.3390/s17112591.
- [6] C. Bartolozzi, L. Natale, F. Nori, and G. Metta, "Robots with a sense of touch," *Nat. Mater.*, vol. 15, no. 9, pp. 921–925, 2016. doi: 10.1038/nmat4731.
- [7] S.-C. Liu and T. Delbruck, "Neuromorphic sensory systems," *Curr. Opin. Neurobiol.*, vol. 20, no. 3, pp. 288–295, 2010. doi: 10.1016/j.conb.2010.03.007.

- [8] H. Landau, "Sampling, data transmission, and the Nyquist rate," *Proc. IEEE*, vol. 55, no. 10, pp. 1701–1706, 1967. doi: 10.1109/JPROC.1967.5962.
- [9] M. Miskowicz, "Send-on-delta concept: An event-based data reporting strategy," *Sensors*, vol. 6, no. 1, pp. 49–63, 2006. doi: 10.3390/s6010049.
- [10] A. E. G. Huber and S.-C. Liu, "On send-on-delta sampling of bandlimited functions," in *Proc. 2017 Int. Conf. Sampling Theory and Applications*, pp. 422–426.
- [11] A. A. Lazar and Y. Zhou, "Reconstructing natural visual scenes from spike times," *Proc. IEEE*, vol. 102, no. 10, pp. 1500–1519, 2014. doi: 10.1109/JPROC.2014.2346465.
- [12] T. Delbruck, "Frame-free dynamic digital vision," in *Proc. Int. Symp. Secure-Life Electronics, Advanced Electronics for Quality Life and Society*, 2008, pp. 21–26.
- [13] A. Khodamoradi and R. Kastner, "O(N)-space spatiotemporal filter for reducing noise in neuromorphic vision sensors," *IEEE Trans. Emerg. Topics Comput.*, to be published.
- [14] D. R. Valeiras, G. Orchard, S.-H. Ieng, and R. B. Benosman, "Neuromorphic event-based 3D pose estimation," *Front. Neurosci.*, vol. 9, pp. 1–15, Jan. 2015. doi: 10.3389/fnins.2015.00522.
- [15] M. Cook, L. Gugelmann, F. Jug, C. Krautz, and A. Steger, "Interacting maps for fast visual interpretation," in *Proc. 2011 Int. Joint Conf. Neural Networks*, pp. 770–776.
- [16] P. Bardow, A. J. Davison, and S. Leutenegger, "Simultaneous optical flow and intensity estimation from an event camera," in *Proc. 2016 IEEE Conf. Computer Vision and Pattern Recognition*, pp. 884–892.
- [17] G. Gallego, J. E. A. Lund, E. Mueggler, H. Rebecq, T. Delbruck, and D. Scaramuzza, "Event-based, 6-DOF camera tracking from photometric depth maps," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 10, pp. 2402–2412, 2018. doi: 10.1109/TPAMI.2017.2769655.
- [18] O. Bichler, D. Querlioz, S. J. Thorpe, J.-P. Bourgoin, and C. Gamrat, "Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity," *Neural Netw.*, vol. 32, pp. 339–348, Aug. 2012. doi: 10.1016/j.neunet.2012.02.022.
- [19] X. Lagorce, G. Orchard, F. Galluppi, B. E. Shi, and R. Benosman, "HOTS: A hierarchy of event-based time-surfaces for pattern recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 7, pp. 1346–1359, 2017. doi: 10.1109/TPAMI.2016.2574707.
- [20] S. Afshar, L. George, J. Tapson, A. van Schaik, and T. J. Hamilton, "Racing to learn: Statistical inference and learning in a single spiking neuron with adaptive kernels," *Front. Neurosci.*, vol. 8, pp. 1–18, Nov. 2014. doi: 10.3389/fnins.2014.00377.
- [21] D. P. Moeys, F. Corradi, E. Kerr, P. Vance, G. Das, D. Neil, D. Kerr, and T. Delbruck, "Steering a predator robot using a mixed frame/event-driven convolutional neural network," in *Proc. 2016 2nd Int. Conf. Event-Based Control, Communication, and Signal Processing*, pp. 1–8.
- [22] J. Anumula, D. Neil, T. Delbruck, and S.-C. Liu, "Feature representations for neuromorphic audio spike streams," *Front. Neurosci.*, vol. 12, pp. 1–12, Feb. 2018. doi: 10.3389/fnins.2018.00023.
- [23] B. Rueckauer and T. Delbruck, "Evaluation of event-based algorithms for optical flow with ground-truth from inertial measurement sensor," *Front. Neurosci.*, vol. 10, pp. 1–17, Apr. 2016. doi: 10.3389/fnins.2016.00176.
- [24] G. C. Carter, "Coherence and time delay estimation," *Proc. IEEE*, vol. 75, no. 2, pp. 236–255, 1987. doi: 10.1109/JPROC.1987.13723.
- [25] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. doi: 10.1038/nature14539.
- [26] I.-A. Lungu, F. Corradi, and T. Delbruck, "Live demonstration: Convolutional neural network driven by Dynamic Vision Sensor playing RoShamBo," in *Proc. 2017 IEEE Symp. Circuits and Systems*, May 28–31, 2017. doi: 10.1109/ISCAS.2017.8050403.
- [27] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *Int. J. Comput. Vis.*, vol. 113, no. 1, pp. 54–66, 2014. doi: 10.1007/s11263-014-0788-3.
- [28] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Front. Neurosci.*, vol. 11, pp. 1–12, Dec. 2017. doi: 10.3389/fnins.2017.00682.
- [29] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: VGG and residual architectures," *Front. Neurosci.*, vol. 13, pp. 1–12, Mar. 2019. doi: 10.3389/fnins.2019.00095.
- [30] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: Opportunities and challenges," *Front. Neurosci.*, vol. 12, pp. 1–18, Oct. 2018. doi: 10.3389/fnins.2018.00774.
- [31] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "EV-FlowNet: Self-supervised optical flow estimation for event-based cameras," in *Proc. Robotics: Science and Systems*, 2018, pp. 1–9.
- [32] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*. Mineola, NY: Dover Publications Inc. 2007.
- [33] S.-H. Ieng, E. Lehtonen, and R. Benosman, "Complexity analysis of iterative basis transformations applied to event-based signals," *Front. Neurosci.*, vol. 12, pp. 1–13, June 2018. doi: 10.3389/fnins.2018.00373.

Signal Processing Foundations for Time-Based Signal Representations

*Neurobiological parallels to engineered systems designed for
energy efficiency or hardware simplicity*



©ISTOCKPHOTO.COM/JUST_SUPER

Neurobiological systems operate at power levels that are unattainable by modern electronic systems while exhibiting broader information processing capabilities for a number of important tasks. A variety of engineered systems designed for energy efficiency or hardware simplicity use time-based signal representations, which share similar mathematical principles with those that arise naturally in biology. In general, time-based signal representations refer to embedding information into the timing, density, or duration of a predetermined, and often bipolar, waveform. In mammalian nervous systems, it is generally accepted that neurons embed information into the timing and firing density of the sudden changes in their membrane potential, or spikes. Similarly, many low-power electronic systems use signal representations that embed information in the timing, repetition frequency, or duration of simple pulse waveforms. Despite their apparent similarities, such signal representations are often studied in different contexts.

This article discusses signal processing foundations for time-based signal representations that use spikes or pulses in a unified framework. Representations of exogenous signals, the inherent sampling mechanisms that arise in this process, reconstructions of these signals from their time-based counterparts, and digital representation of spike- and pulse-based signals are specifically discussed.

Introduction

Time-based signal representation refers to embedding information into the timing, density (repetition frequency), or duration of a waveform rather than, for example, its amplitude. Which waveform is chosen and how it is used to represent information may differ among systems, giving rise to a set of interesting properties. Neurobiological systems in general, and mammalian nervous systems in particular, exhibit time-based signal representations by modulating the timing and density of action potentials. Action potentials, more commonly known as *spikes*, are millisecond-width signals of millivolt-scale electrochemical potential that propagate along neuronal cell membranes to facilitate rapid neuron-to-neuron communication

over distances in the nervous system ranging from millimeters to meters. Signal representations beyond action potentials also appear in biology. For instance, plateau potentials arise in some nonmammalian nervous systems [1]. As indicated in Table 1, spikes are asynchronous, continuous-time signals $f(\cdot)$, of short duration, whose temporal support is often of negligible importance. In general, spikes are mathematically described using the Dirac delta function $\delta(t)$. This convention will be used in this article.

In circuit applications, synchronous, bipolar (rectangular) pulses are often used when representing signals in the time domain, thanks to the simplicity of their generation and their well-defined timing and duration. In practice, pulses take a finite amount of time to transition between states, denoted by $\sigma(t)$ in Table 1. However, pulses are often modeled as transitioning between levels instantaneously, as indicated by the use of unit step function $u(t)$ in Table 1. This makes it straightforward to encode information into the timing of one or more of the edges (relative to a synchronous clock or carrier signal) or the duration of such pulse shapes, as well as in the density of occurrences of such pulse shapes over a longer duration. This article discusses signal processing foundations for time-based signal representations and points to connections between spike- and pulse-based methods.


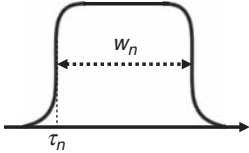
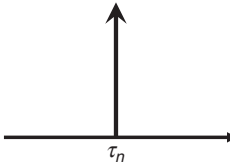
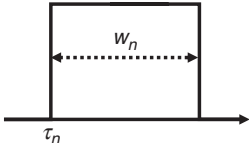
Time-based signal representations span a large application space from brain- and biology-inspired computation models to robust circuits-and-systems applications. IBM's neurosynaptic computer chip, TrueNorth, uses spike-based signal representations in a non-von Neumann computing architecture [2]. Pulse-width modulation (PWM) is a bipolar, quasi-periodic signal representation that embeds samples of the amplitude of an input signal into the widths of modulating pulses. It is used in applications ranging from optical data storage [3], [4] to audio amplification [5], analog-to-digital conversion [6], and power

conversion [7]. Pulse-position modulation (PPM) represents an underlying signal with the relative timing of short pulses in an interval. It is often employed in optical communication systems due to its multipath interference rejection [4].

Another time-based signal representation, pulse-density modulation (PDM), is particularly common in audio applications [8] and represents information through the repetition frequency, or density, of the modulating pulses, similar to densely timed spikes, or bursts, in neural communications [9]. In sensing applications in general, and those employing optical sensors in particular, sensors are often known to produce photoelectrical spikes at a rate that is proportional to the intensity of the incoming light, induced by the incidence of photons on a photosensitive substrate. As such, the resulting temporal waveform encodes the intensity profile of the light signal in the density of such photon-induced arrivals, which are often modeled stochastically as Poisson, similar to the neuronal spike trains [10]. Generalizations from dc signaling to oversampled representations of band-limited signals have so dominated the microprocessor application space, in many application-specific microprocessors, analog output pins have been completely replaced by their PWM or PDM counterparts. Similarly, modern microelectromechanical system microphones, which are pervasive in the Internet of Things and devices for far-field audio capture such as smart speakers, output PDM signals nearly exclusively. Figure 1 illustrates several time-based signal representations, along with some of their applications and relationships between spike/burst modulations and pulse-based modulations, including PWM, PPM, and PDM.

How information is embedded and what information is embedded into the timing, density, or duration of waveforms that comprise a time-based signal representation give rise to a set of interesting properties and lay a foundation for a unified framework for signal processing via biology- and systems-inspired

Table 1. Spike- and pulse-based signal representations.

| | Spike | Pulse |
|-------------------------------|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| Example of the waveform shape |  |  |
| Functional form | $f(t - \tau_n)$ | $\sigma(t - \tau_n) - \sigma(t - \tau_n - w_n)$ |
| Abstraction |  |  |
| Functional form | $\delta(t - \tau_n)$ | $u(t - \tau_n) - u(t - \tau_n - w_n)$ |
| Carrier of information | Timing, density of occurrence | Timing of edges, density, and duration of pulses |
| Synchronicity | Usually asynchronous | Usually synchronous, quasi-periodic |

signal representations. Specifically, the spike and burst random processes generated by modulating the intensity function of a Poisson process and random pulse processes describing the same underlying signal of interest can be related through differentiation or integration up to scaling and offsets. Deterministic models for spike- and pulse-based representations also exhibit similar information-embedding techniques that can be used to represent, for example, finite-energy band-limited signals.

A unified framework for time-based signal representations

A distinctive aspect of time-based signal representations is the apparent simplicity with which they represent information through the relative placement of a given waveform shape potentially along with simple temporal scaling of the nominal waveform. Formally, time-based signals that represent information in the timing, density, or duration of nominal waveforms have the following form:

$$q(t) = \sum_{n \in \mathbb{Z}} f_n(t - \tau_n). \quad (1)$$

Here, nominal waveforms $f_n(\cdot)$ are functions of compact support and τ_n denotes the placement of the modulated pulse shape along the time axis. For synchronous methods, this placement generally lies within a given pulse interval, $\tau_n \in [nT, (n+1)T]$, for a given interval duration T . The nominal pulse shapes $f_n(\cdot)$ do not vary, or vary negligibly, in amplitude but they often have varying temporal support that lies in a predetermined compact set. In the case of density-based representations, $f_n(\cdot)$ could correspond to the superposition of nominal pulse shapes that ideally do not overlap temporally.

Figure 2 summarizes the time-based signals of interest that embed information about an exogenous signal into the set of pairs $\{f_n, \tau_n\}$: Spike-based signals employ firing instances τ_n of spikes, which are commonly modeled as Dirac delta functions as $f_n(t) = \delta(t)$ for all n . Similarly, bursts of spikes are modeled as a collection of spikes that are densely spaced in time, $f_n(t) = \sum_{i=1}^{d_n} \delta(t - \tau_{ni})$ for some density variable d_n and spike distribution $\{\tau_{ni}\}_{i=1}^{d_n}$. Ignoring synchronicity, functional forms of PPM and PDM mirror those of spike and burst modulation with a slight modification. $\delta(t)$ is replaced by a

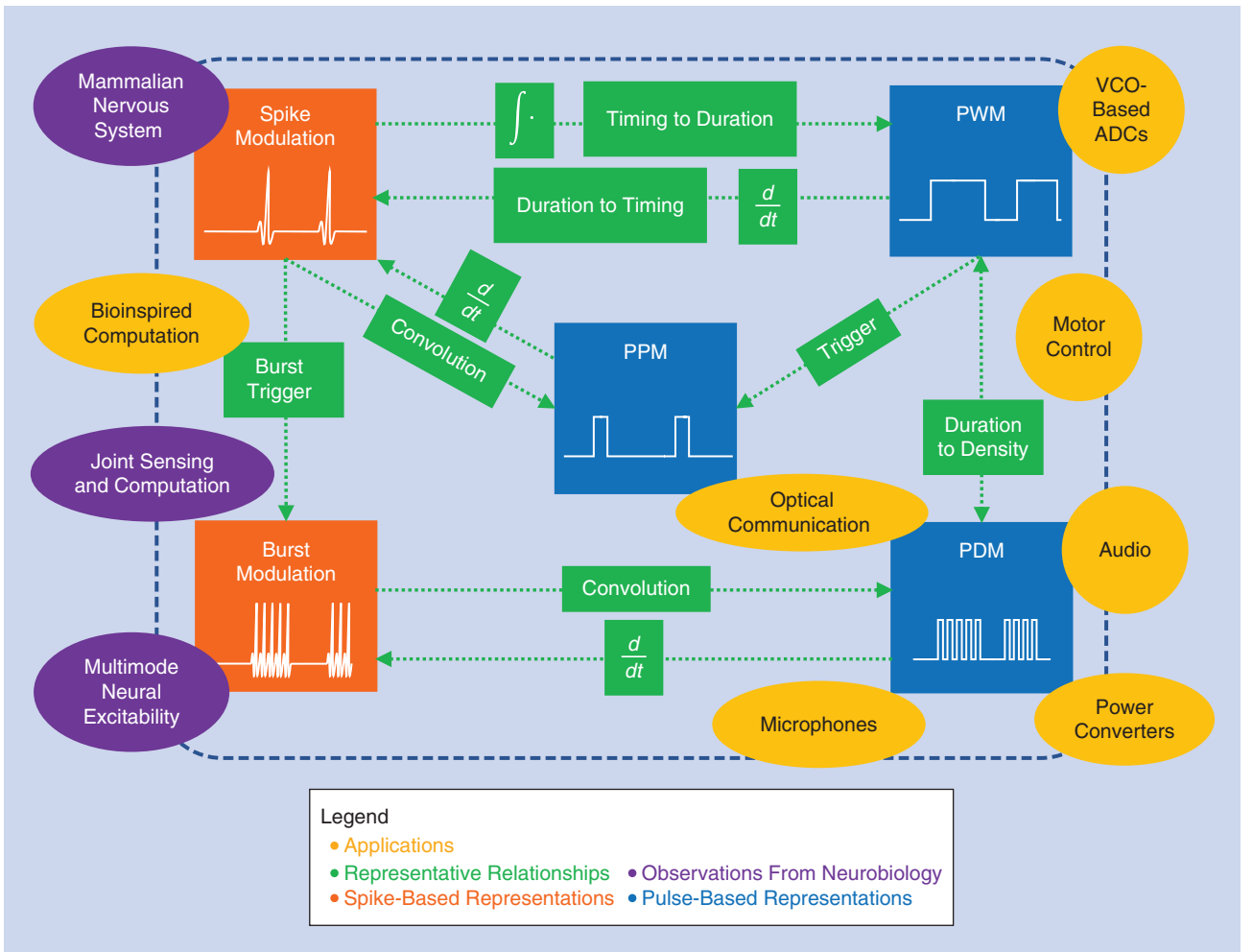


FIGURE 1. The application space of time-based signal representations and transformations between timing, duration, and density as well as between spike- and pulse-based representations. For simplicity, synchronicity and the existence of a carrier, or pulse interval, is not shown. ADC: analog-to-digital converter; VCO: voltage-controlled oscillator.

short pulse $\rho(t) = u(t) - u(t - w)$, where $u(t)$ is the unit step function and w is the pulse duration that is commonly short compared with the reciprocal of the bandwidth of the exogenous signal of interest. Finally, PWM signals, which embed information into the duration of the nominal pulses, exhibit the form $f_n(t) = u(t) - u(t - w_n)$. Each of these representations can embed information in the timing τ_n , duration, or density of successive nominal waveforms $f_n(t)$.

While these time-based representations may be used to encode information that originates at the encoder itself, we consider the use of such representations to both sample and encode an exogenous signal of interest, $x_o(t)$, as these are the applications of most common use for these methods. As discussed in the “Introduction” section, whether the exogenous signal is deterministic or stochastic depends on the application. While most audio and electromechanical

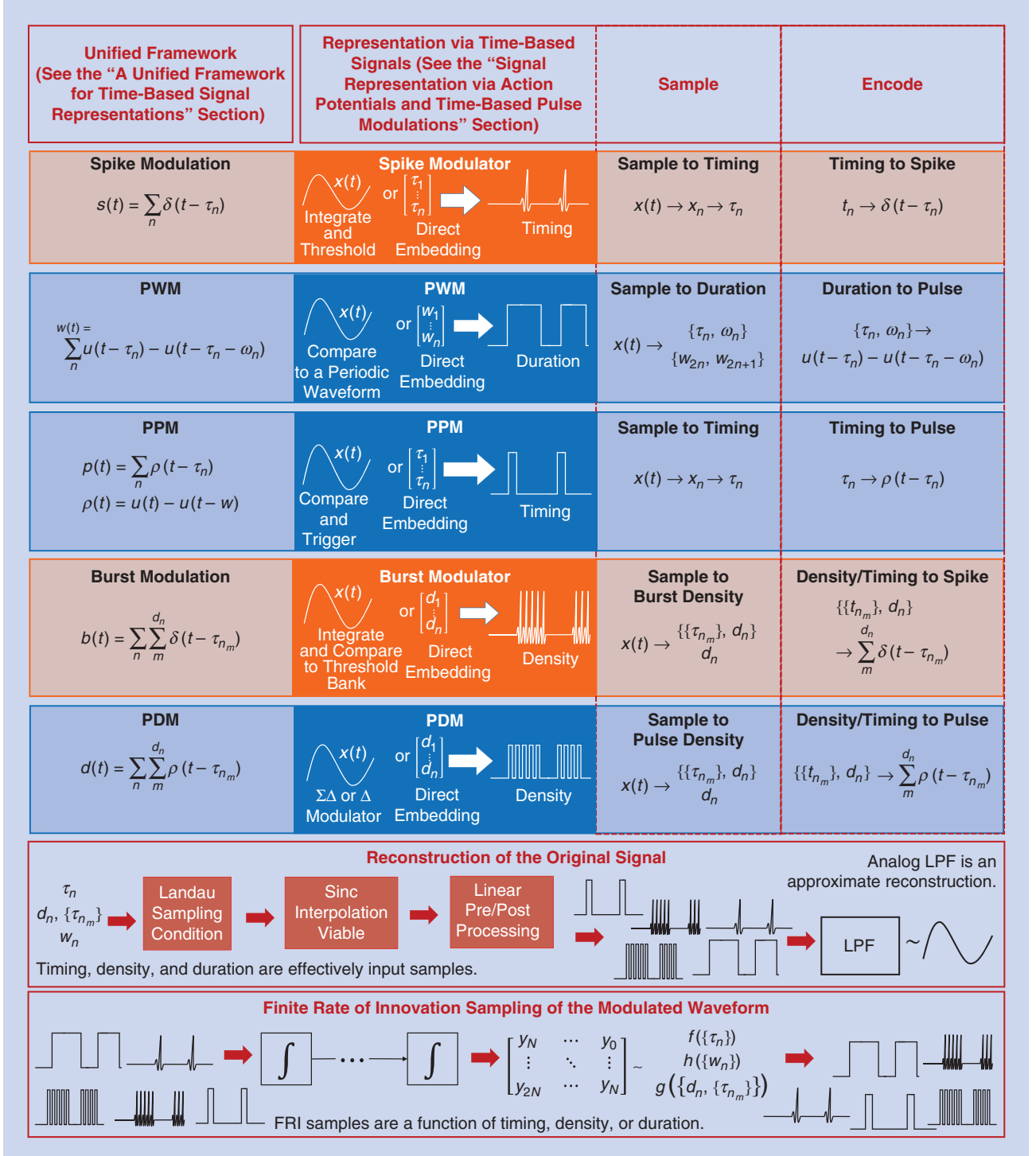


FIGURE 2. An overview of the representation, sampling, and reconstruction of time-based signals. LPF: low-pass filter.

applications are governed by deterministic design rules, sensing applications in particular are often inherently stochastic. When the exogenous signal is modeled as controlling the rate parameter of a Poisson arrival process (as in the optical sensing example), a conceptually useful mathematical relationship connects pulse- and spike-based representations and is explored next.

A Poisson model for spike- and pulse-based signal representations

The Poisson spiking model from computational neuroscience asserts the independent spiking hypothesis that spikes are released approximately independently with exponentially distributed first-order arrival times, resulting in a Poisson arrival process [9], [11], [12]. Here, we provide a stochastic differential equation (SDE) framework for the Dirac spiking model and make connections to time-based stochastic pulse modulators. Consider a Poisson counter N_t of fixed arrival rate λ . With initial condition $N_0 = 0$, the Poisson counter N_t has the average of $\mathbb{E}[N_t] = \lambda t$, which motivates the notion of λ as the rate. The Poisson process model can also be derived from the notion that in a sufficiently small time interval of length ΔT , the probability of an arrival is proportional to ΔT , i.e.,

$$\mathbb{P}(N_{t+\Delta T} - N_t = 1) = \lambda \Delta T, \quad \mathbb{P}(N_{t+\Delta T} - N_t = 0) = 1 - \lambda \Delta T.$$

Define a spike-train S_t generated from the Poisson counter N_t as

$$S_t = dN_t. \quad (2)$$

Differentiation is understood to produce positive Dirac delta functions at the points of discontinuity, i.e., at the arrival times of the Poisson process.

Similar to the Poisson spiking model in (2), it is possible to represent a signal N_t using a time-based random pulse modu-

lator. For notational simplicity, we use a $\{-1, 1\}$ bimodal random pulse modulator, denoted by P_t :

$$dP_t = -2P_t dN_t. \quad (3)$$

As long as the initial condition P_0 is in $\{-1, 1\}$, the modulating signal P_t is in $\{-1, 1\}$ almost always (up to a set of measure zero over real numbers, on which the random process P_t is defined). As shown in Figure 3, P_t transitions between states when the underlying Poisson counting process N_t jumps.

The number of jumps is determined by the rate λ of the Poisson counter N_t . If the Poisson counting process N_t has a fixed rate λ , P_t becomes a square wave with duty cycle in expectation given by $2/\lambda$, i.e., the expected second-order interarrival time for the Poisson process. The instantaneous duty cycle, the time between positive edges in the pulse process, would have a second-order Erlang distribution, $f_i(t_0) = \lambda^2 t_0 e^{-\lambda t_0}$, $t_0 \geq 0$. When the rate of the Poisson counting process is a finite-energy, band-limited deterministic function $x(t)$, the solution P_t to the SDE in (3) embeds the input signal into the transition times via the counting process

$$\begin{aligned} \mathbb{P}(N_{t+\Delta T} - N_t = 1) &= x(t)\Delta T, \quad \mathbb{P}(N_{t+\Delta T} - N_t = 0) \\ &= 1 - x(t)\Delta T, \end{aligned}$$

such that the number of arrivals in the interval (t, τ) has the distribution

$$\mathbb{P}(N_\tau - N_t = k) = m(t, \tau)^k e^{-m(t, \tau)} / k!, \quad m(t, \tau) = \int_t^\tau x(s) ds.$$

Recovery of the signal $x(t)$ from the random spike train or random PWM signals would correspond to the recovery (or estimation) of the signal incident on the sensor modulating its firing rate, such as an optical signal incident on a p-i-n or avalanche photo diode.

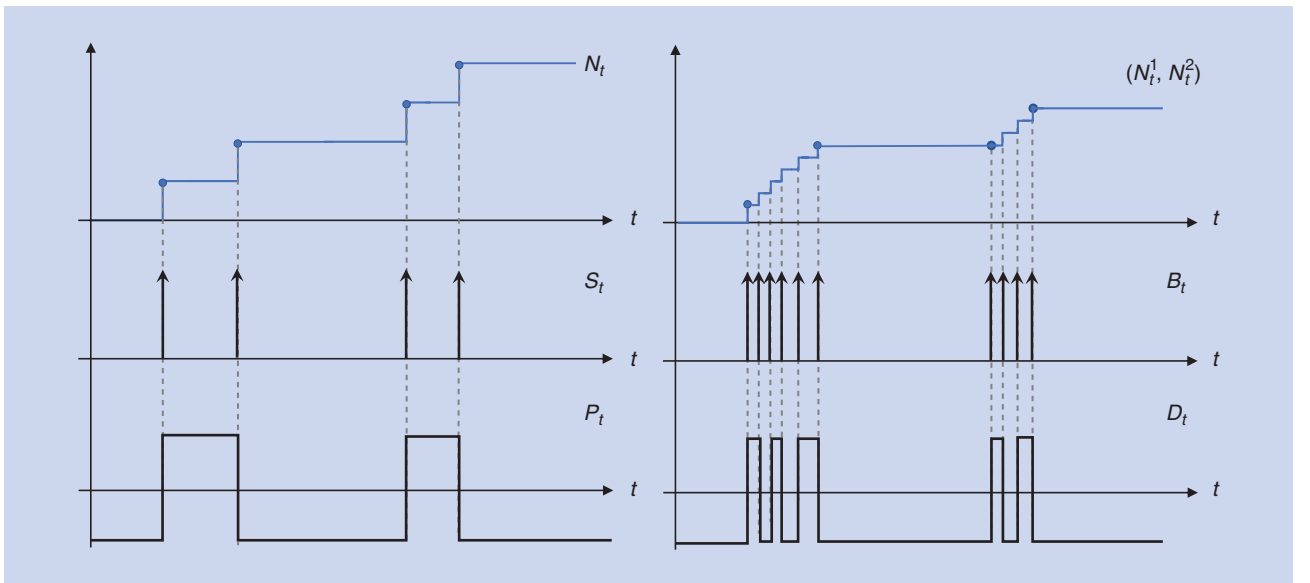


FIGURE 3. The sample paths from a Poisson counter N_t and the corresponding spike train S_t and random PWM P_t , a compound Poisson counter and the corresponding burst train B_t and random PDM D_t .

A stochastic burst model involves a combination of two Poisson counting processes. The first, N_t^1 , models when a burst happens and the second, N_t^2 , determines how many spikes there are at each burst [10]. This might be a suitable model for an optical detector that uses a photomultiplier tube, generating a cascade of photoelectrons for each incident photon on the detector. A Poisson-burst neuron B_t is governed by the following set of SDEs:

$$\begin{aligned} B_t &= (1 + P_t)dN_t^2 \\ dP_t &= -2P_t dN_t^1. \end{aligned} \quad (4)$$

Similar to (3), P_0 is in $\{-1, 1\}$. Typically, the burst frequency (the number of spikes at each burst) is much higher than the frequency at which the burst are fired. Formally, rates $\{\lambda_1, \lambda_2\}$ that correspond to the Poisson counting processes N_t^1 and N_t^2 satisfy $\lambda_1 \ll \lambda_2$. Equation (4) can be interpreted as follows. Arrivals from the first Poisson process signal the start and stop times of each burst interval, essentially throwing a switch, turning the arrivals from the second process on and off. During this burst interval, arrivals from the second counting process are passed through to B_t , while arrivals from the second process are zeroed out during the off phase.

Stochastic pulse-density modulators follow a similar set of SDEs. Reminiscent of the relation between (2) and (3), SDEs that govern a pulse-density modulated process D_t mirror that of the burst model

$$\begin{aligned} dD_t &= -(1 + P_t)D_t dN_t^2 + \frac{(1 - P_t)}{2} \\ dP_t &= -2P_t dN_t^1. \end{aligned} \quad (5)$$

Here, $(1 - P_t)/2$ is a correction term that defines the state (either +1 or -1) in which the PDM process rests in the no-burst period. Figure 3 illustrates pulse-density, pulsewidth, spike, and burst-modulated processes that are driven by Poisson jump processes (N_t^1, N_t^2). Poisson spiking, bursting, and pulsing models in (2)–(5) indicate that spiking and pulsing signal representations are related through integration or differentiation up to linear (affine) preprocessing. Spiking and bursting processes are derivatives of the underlying jump processes, where pulsing processes are their integrals over the Galois field.

How would the modulated signal differ between pulse- and spike-based representations when the modulating signal $x_o(t)$ is deterministic instead? Would the spiking/pulsing instances coincide for different pulsing/spiking machines or modulators? Is it possible to represent any finite-energy, band-limited signal in a lossless manner using such time-based representations? We address these questions next.

Deterministic models for time-based representations

A distinctive aspect of time-based pulse modulators, and of many biological nervous systems, is the use of purely analog pipelines. Action potentials are continuous-time signals with varying functional properties [9], facilitating sensing, communication, and information processing through the nervous system [13]. Furthermore, the input signals for

time-based applications are rarely discretized (sampled) for system simplicity and the modulating signals are continuous time by construction [6], [14], [15]. As discussed in the “Reconstruction From Time-Based Representations” section, even reconstruction of the original input signals from their time-based counterparts can be carried out, approximately, via analog LPF.

Some of the earliest uses of time-based signal representations included PWM signaling to convey constant or nearly constant signal values for electromechanical applications, such as motor drivers, dc power circuits, or other cases for which long sequences of duty-cycled square-wave signals could simply be integrated to recover the desired dc values. As discussed in the Introduction, time-based signals have become prominent tools for power-constrained signal representation for a variety of low-bandwidth applications in general, not limited to conveying dc values alone.

As long as the exogenous signal $x(t)$ is band-limited and finite-energy, which necessarily implies being continuous and bounded [16], it can be reliably represented by a time-based signal [15]. Therefore, let us consider representation of a finite-energy, band-limited continuous-time signal $x(t)$, via time-based signal representations.

An idealized spike train has the functional form [17]

$$s(t) = \sum_{n \in \mathbb{Z}} \delta(t - \tau_n). \quad (6)$$

Here, the sequence $\{\tau_n\}_{n \in \mathbb{Z}}$ of spiking instances represents (implicitly obtained) samples of the underlying signal $x(t)$ and \mathbb{Z} denotes the integers. Incidentally, the functional form in (6) extends to PPM directly

$$p(t) = \sum_{n \in \mathbb{Z}} u(t - \tau_n) - u(t - \tau_n - w) = \sum_{n \in \mathbb{Z}} \rho(t - \tau_n), \quad (7)$$

where $u(\cdot)$ is the unit step function and w is a fixed duration that is commonly short compared to the implicit sampling period T that corresponds to the reciprocal of the bandwidth of the modulating signal $x(t)$: $w \ll T$.

PWM is a quasi-periodic bipolar waveform that represents band-limited finite-energy signals by encoding samples of the input signal into the duration or width of the modulated pulses. It has the functional form

$$\begin{aligned} w(t) &= \sum_{n \in \mathbb{Z}} u(t - w_{2n}) - u(t - w_{2n+1}) \\ &= \sum_{n \in \mathbb{Z}} u(t - \tau_n) - u(t - \tau_n - \omega_n). \end{aligned} \quad (8)$$

Here, $\{w_n\}$ is the sequence of rising and falling-edge instances. We take the convention of $\{w_{2n}\}$ denoting the rising-edge instances and $\{w_{2n+1}\}$ denoting the falling-edge instances. A PWM signal $w(t)$ represents the samples x_n from the input signal $x(t)$ with pulses of widths $\omega_n = w_{2n+1} - w_{2n}$ fired at $\tau_n \in \mathcal{T}_n$.

Action potentials represent information about their densities as well as timings [9], [11], [18]. Collections of closely spaced spikes resulting from sudden discharges of electrons due to higher levels of neural excitations are commonly

referred to as *bursts*. The integrated fire-or-burst neuron model has what is called a *burst mode* when it represents a signal via such densely spaced spikes [18], [19]. The ability to make transitions between different time-based signal representations is called *multimodality*, and the integrated fire-or-burst neuron is the simplest model that captures this behavior. Even though the form in (6) also captures bursting signals, it is more intuitive to construct a form that preserves the quasi-periodicity explicitly:

$$b(t) = \sum_{n \in \mathbb{Z}} \sum_{m=1}^{d_n} \delta(t - \tau_{nm}). \quad (9)$$

Here, d_n represents the input sample that corresponds to the sampling interval \mathcal{T}_n and the timing $\{\tau_{nm}\}$ of the spikes are contained in a fixed time window of duration $T_B < T$.

Information embedding into firing density, or repetition frequency, also appears in PDM:

$$d(t) = \sum_{n \in \mathbb{Z}} \sum_{m=1}^{d_n} \rho(t - \tau_{nm}), \quad (10)$$

where $\rho(\cdot)$ is a pulse of fixed duration, T_D . Sigma-Delta ($\Sigma\Delta$) modulation is a common way to create pulse-density modulated signals. An analog input signal $x(t)$ is integrated and quantized. This quantized estimate is fed back and subtracted from the input at rate $1/T_D$. The resulting output of the quantizer is a type of PDM signal. The original signal $x(t)$ is often reconstructed by simply LPF the PDM waveform. For Δ modulation, the signal $x(t)$ is directly quantized and the integral of the quantized signal is subtracted from the input, again at rate $1/T_D$. The resulting quantizer output is also a PDM waveform and the original signal is often estimated by integrating this PDM signal followed by an LPF.

In deterministic models, $\{\tau_n, w_n, d_n\}$ obey similar notions of quasi-periodicity. Timing instances τ_n of spikes or pulses rest in what are called *sampling intervals* $\mathcal{T}_n: \tau_n \in \mathcal{T}_n$ and the duration of \mathcal{T}_n is called the *sampling period*, which is uniform over all samples, $|\mathcal{T}_n| = T$ for every sampling interval [15]. Furthermore, every PWM pulse happens in a single sampling interval w_{2n}, w_{2n+1} are in \mathcal{T}_n and $w_{2n+1} - w_{2n} < T$. Similar notions extend to burst modulation and PDM. How the timing, density, and duration of pulses are generated are addressed next.

Signal representation via action potentials and time-based pulse modulations

Encoding an analog signal into a time-based signal representation can be accomplished with relatively simple hardware, yet the resulting mapping can be difficult to describe and analyze mathematically. The sample-and-encode principle is a way to (at least conceptually) simplify the modeling and analysis of time-based signal representations. Many such methods directly map an analog signal $x(t)$ into a time-based signal such as PWM by using little more than a reference signal and a comparator. However, it is more convenient to think of there being a sampling phase that takes samples of the analog signal at some (possibly irregular, signal-dependent) time intervals, followed by an encoding phase

that maps these samples into the timing of the pulse-based waveform, as illustrated in Figure 2. The sample-and-encode principle provides the intermediate sequence of mappings

$$x(t) \rightarrow x_n = x(\tau_n) \rightarrow \{d_n, \tau_n\} \rightarrow q(t),$$

where x_n are input samples, $\{d_n, \tau_n\}$ are the pulse shapes and timing that represent the input sample, and $q(t)$ is the resulting time-based waveform, as shown in (1). For example, the timing of spikes and PPM pulses, the frequency of bursting spikes or PDM pulses, and the duration of PWM pulses are functions of the input samples. The sample-and-encode principle enables a straightforward comparison of some of the properties of different spiking and pulsing representations. In particular, we separately compare the sampling and encoding steps of a time-encoding machine to those of pulse modulators and investigate the transformations among the corresponding samples and waveforms.

Next, we introduce and compare sample-and-encode principles that govern the generation of spike trains and PWM signals. We first introduce time-encoding machines $x(t) \rightarrow s(t)$ and PWM $x(t) \rightarrow w(t)$. Then, we discuss transformations between spike trains and PWM signals $s(t) \leftrightarrow w(t)$. We further address the relationship between spiking instances τ_n and pulsewidths $\omega_n = w_{2n+1} - w_{2n}$.

Time-encoding machines and PWMs

The integrate-and-fire (I&F) neuron model is one of the least computationally expensive models for an action potential neuron [18]. In [17] and [20], Lazar introduces a time-encoding machine model to study a possible means by which band-limited signals might be encoded in neuronal spike trains and the conditions under which they can be losslessly recovered. The idea behind the I&F model is that a neuron decides to fire a spike when an excitation has accumulated sufficient membrane potential.

Figure 4 illustrates a time-encoding machine operating over a single sampling interval [17], which is a slight modification to Lazar's model (which introduced a constant refractory period) to include a variable refractory period. This enables us to guarantee that a single spike occurs within each interval \mathcal{T}_n by allowing the integrator to periodically reset. Such a time-encoding machine generates the spike-train signal $s(t)$ that represents the analog signal $x(t)$ through the following steps.

- 1) *Preprocessing*: A bias is introduced to force input signal to be strictly nonnegative. $\hat{x}(t) = x(t) + \|x\|_\infty$.
- 2) *Integration*: The preprocessed signal $\hat{x}(t)$ has a monotonically increasing integral $r(t)$.
- 3) *Thresholding*: Comparing the integral to a fixed threshold δ ensures that a transition happens later when the input signal is lower and earlier when the input amplitude is higher, leading to sampling instance τ_n :

$$\tau_n = \min_{\tau \in \mathcal{T}_n} \left\{ \tau: \int_{nT}^{\tau} x(t) dt + \|x\|_\infty (\tau - nT) = \delta \right\}. \quad (11)$$

- 4) *Derivative*: A spike is fired when a state transition happens. Similar to the "A Poisson Model for Spike- and Pulse-Based Signal Representations" section, this is a left derivative.

Preprocessing, integration, and thresholding steps constitute the sampling phase of a time-encoding machine, in which the thresholding and derivative steps yield the encoding into a waveform. In the system proposed in [17], the thresholding and derivative operations are carried out jointly via a noninverting Schmitt trigger. The sequence of spiking instances in (11) raises the following several questions:

- 1) *Existence*: Does τ_n necessarily exist in every sampling interval \mathcal{T}_n ?
- 2) *Uniqueness*: Is there a unique τ_n corresponding to $x(t)$ over \mathcal{T}_n ?
- 3) *Representation*: Does the sequence $\{\tau_n\}$ represent $x(t)$ in a lossless fashion? Equivalently, does there exist a mechanism that reconstructs $x(t)$ from $\{\tau_n\}$ to within a set of measure zero?

Such questions are addressed in a number of nonuniform sampling problems, for example, [15], [17], and [21] and the references therein. Spiking instances $\{\tau_n\}$ are also the sampling instances that represent the underlying signal $x(t)$. For a fixed refractory period with bias in Figure 4 larger than $\|x\|_\infty$, as long as $\Omega_0 \leq g(\delta, T)$ for $g(\cdot)$ as given in [17], a time-encoding machine represents the finite-energy, band-limited signal $x(t)$ in a lossless manner. Furthermore, there exists a unique spike at time $\tau_n \in \mathcal{T}_n$, representing an input sample via (11). Finally, as long as (δ, T) are chosen to satisfy $\Omega_0 \leq f(\delta, T)$ [17], there exists a lossless sinc interpolation mechanism to reconstruct the original signal via $\{\tau_n\} \rightarrow x(t)$, through applying the Nyquist sampling theorem.

Observe that without the derivative operation in Figure 4, a time-encoding machine encodes information into pulses rather than spikes, yielding a PWM signal. Figure 5 illustrates different approaches to PWM generation for representing finite-energy, band-limited signals. A natural abstraction applies the sample-and-encode principle $x(t) \rightarrow x_n \rightarrow w_n \rightarrow w_{\text{GEN}}(t)$. Here, $x(t) \rightarrow x_n$ is the *sampling* part via continuous-to-discrete (C/D) time converter, $x_n \rightarrow w_n$ is commonly an affine mapping from samples to pulsewidths and $w_n \rightarrow w_{\text{GEN}}(t)$ is the pulse generation. Collectively, $x_n \rightarrow w_{\text{GEN}}(t)$ is also called the *encoding* part [15]. So-called uniform sampling PWM is a conceptually simple, though overly hardware-complex, means for PWM gener-

ation that first uniformly samples the analog signal producing samples $x_n = x(nT)$ and then converts them into pulsewidths, often by means of comparison with a linear ramping saw-tooth signal, yielding $x(t) \rightarrow x_n \rightarrow w_{\text{US}}(t) = (x_{\text{ZOH}}(t) > r_{\text{sync}}(t))$. A zero-order hold (ZOH) signal $x_{\text{ZOH}}(t)$ is generated from uniformly spaced input samples $x_n = x(nT)$ and is dynamically compared to a reference signal synchronized to the sample-and-hold block $r_{\text{sync}}(t)$ to generate uniformly sampled (US) PWM signal $w_{\text{US}}(t)$. Such a PWM signal directly embeds $x(nT)$ into $w_{2n+1} - w_{2n}$, allowing conventional sinc-interpolation to reconstruct the original signal from samples mapped back from pulsewidths $\omega_n \rightarrow x_n = x(nT)$.

In terms of power consumption and circuit area to implement such externally clocked, C/D-based uniform sampling PWM generators or time-to-digital converters (TDCs) to measure pulsewidths [22] it is not efficient [22], [23]. Therefore, a technique referred to as *natural sampling* is implemented far more commonly to generate PWM signals that represent finite-energy, band-limited signals. Such designs compare the input signal $x(t)$ directly to a periodic reference signal $r(t)$ dynamically over time, taking the comparator output as the PWM representation. This results in nonuniformly spaced input samples embedded in the timing of pulses, $w_{\text{NS}}(t) = [x(t) > r(t)]$. The functional properties of $r(t)$ not only determine the pulse-orientation of the resulting signals $w_{\text{NS}}(t)$, but also determine whether the representation is lossless [15]. A natural sampling PWM generator is a purely analog generator that represents $x(t)$ via natural sampling instances ω_n

$$\begin{aligned} \omega_n &= \{w_{2n_i} - w_{2n_i+1} : x(w_{2n_i}) = r(w_{2n_i}), x(w_{2n_i+1}) \\ &= r(w_{2n_i+1}), \forall w_{n_i} \in \mathcal{T}_n\}. \end{aligned} \quad (12)$$

The sampling instances defined in (12) raise questions regarding properties similar to those raised for spike trains for the definition in (11), about existence, uniqueness, and reconstruction. As investigated in [15], a lossless natural sampling PWM signal has a unique pulse of width ω_n that corresponds to a non-uniformly spaced input sample x_n . Furthermore, there exists a reconstruction mechanism $\omega_n \rightarrow x(t)$ that is lossless in the Landau sense [24]. Time-encoding machines and PWM can ensure

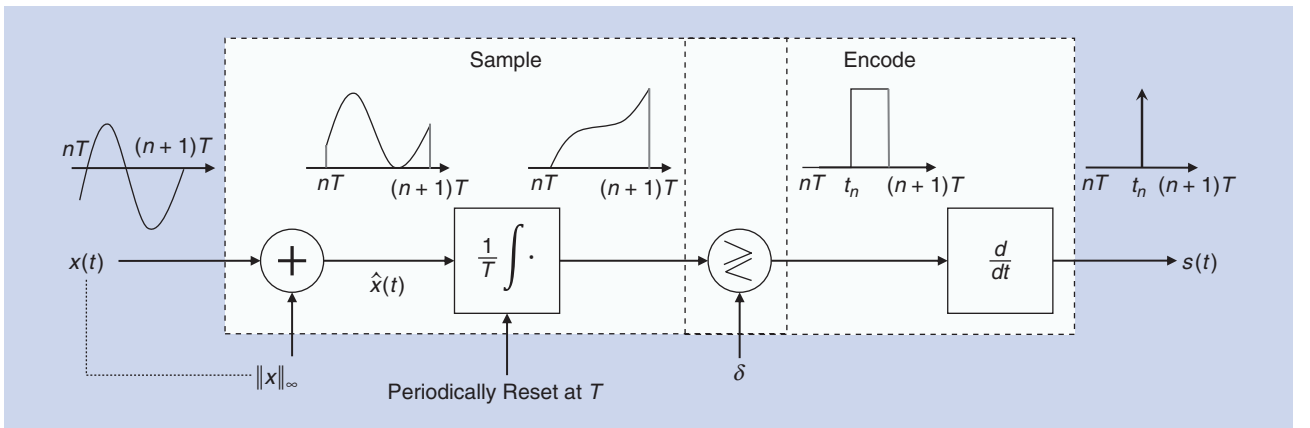


FIGURE 4. A time-encoding machine of an I&F neuron with variable refractory period.

lossless time-based representations of finite-energy band-limited signals through different means, effectively corresponding to different sampling schemes [15], [17]. We next investigate transformations $s(t) \leftrightarrow w(t)$ between lossless spike train representations and PWM signals and consider some of their spectral properties. The implicit sampling instances that arise in a time-encoding machine and those of natural sampling PWM are not necessarily the same. However, under the conditions listed in [15] and [17], both $\{\tau_n\}$ and ω_n can represent $x(t)$ in a lossless manner.

Despite a simple transformation between these signals, their spectral properties differ. Given $\{\tau_n\}$, the Fourier transform of $s(t)$, denoted by $S(j\Omega)$, is

$$S(j\Omega) = \sum_{n \in \mathbb{Z}} e^{-j\Omega\tau_n}, \quad (13)$$

while the frequency-domain representation of a trailing-edge PWM signal is given by

$$W(j\Omega) = \sum_{n \in \mathbb{Z}} \int_{nT}^{\tau_n} e^{-j\Omega t} dt. \quad (14)$$

For sufficiently high oversampling rates, $\tau_n - nT \ll 2\pi/\Omega_0$, where, Ω_0 is the bandwidth of $x(t)$, (14) is approximately $X(j\Omega)$ with mostly out-of-band harmonic distortion. As such, LPF reconstruction from PWM representations becomes asymptotically lossless in the oversampling factor [25]. For I&F neurons with a fixed refractory period, sinc-interpolation of

weighted spikes is proposed to reconstruct the original signal [17]. A general framework for sinc-interpolation from nonuniformly spaced samples is given in [21].

While PWM signals generated by time-encoding machines or those generated by natural sampling PWM or uniform sampling PWM may have different sampling instances, they all represent the underlying signal $x(t)$ faithfully, losslessly, with each having a different nonlinear reconstruction mechanism. However, they also provide asymptotically lossless LPF reconstruction. We gain intuition as to why this happens by noting that although the sampling instances of each of these methods differ, the band-limited nature of $x(t)$ forces the resulting sample values to remain close. Since band-limited, finite-energy signals have finite variation, we have [15]

$$|x(t) - x(t + \tau)| \leq \tau \Omega_0 \|x\|_2 \sqrt{\frac{\Omega_0}{3\pi}}. \quad (15)$$

Here, $\|x\|_2 = \sqrt{\int |x(t)|^2 dt}$, where $\|x\|_2^2$ is the energy of $x(t)$. The bound in (15) necessarily constrains the natural samples ω_n as well as the spiking instances τ_n as they are monotonically increasing functions of the input amplitude. Therefore, the sampling instances τ_n from (11) and the pulsewidths ω_n from (12) must lie in a common subset of the sampling interval \mathcal{T}_n , parametrized by $(\Omega_0, T, \|x\|_\infty, \delta)$. Consequently, spike trains that are generated by I&F neurons and natural sampling PWM signals are capable of representing finite-energy band-

limited signals in a lossless manner, while employing different waveforms to incorporate closely contained samples. A similar relation exists between PDM and neural responses from the integrate-and-fire-or-burst (I&FB) model in the burst mode.

Burst modulation and PDM

When the burst mode is enabled, the time-encoding machine in Figure 6 generates a burst of spikes that represent the input signal via d_n , which corresponds to the number of levels crossed [18]. The portion of the integrated signal that exceeds the threshold, $r_{\text{sup}}(t)$ is dynamically compared to a set of levels producing a staircase signal of varying jump instances $\{\tau_{nm}\}$, which triggers the spikes that form a burst.

Burst instances $\{\tau_{nm}\}$ are not necessarily uniformly spaced, neither among bursting spikes nor among sampling intervals. Depending on the distribution of the level sets, $\{\tau_{nm}\}$ might arbitrarily occupy the burst range of duration T_B . Given the condition for enabling burst mode and the distribution of level sets, bursting density characterizes the input signal objectively as $x(t) \rightarrow d_n$. For

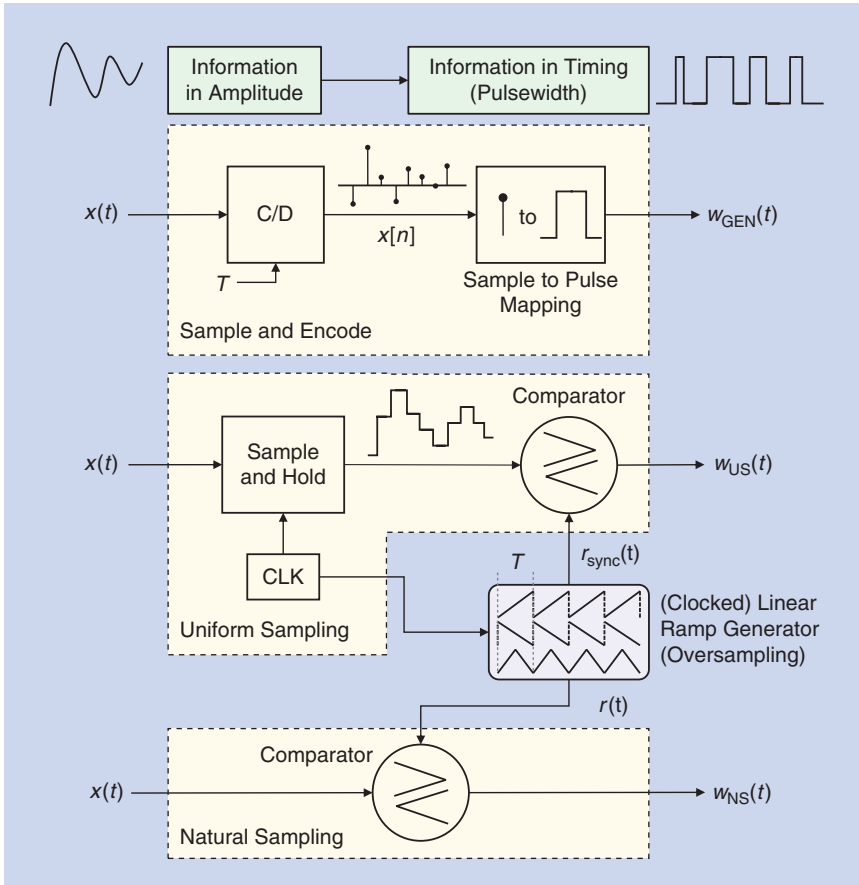


FIGURE 5. The PWM from concept to implementation.

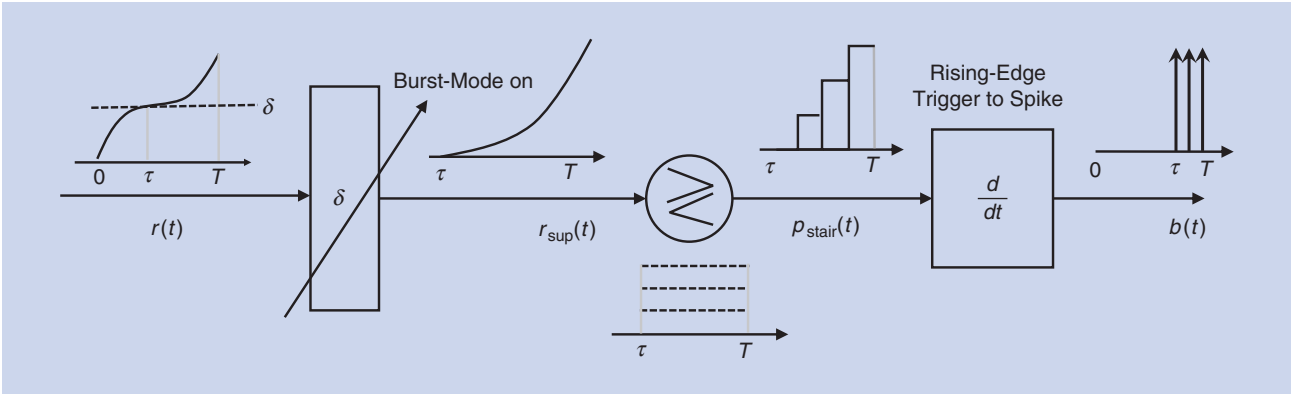


FIGURE 6. I&FB neuron in burst mode.

instance, in the case of bursting with equally spaced spikes for excitations exceeding the threshold, the burst density is given by

$$d_n = \left\lfloor \frac{\|x\|_\infty T + \int_{nT}^{(n+1)T} x(t) dt - \delta}{\delta_B} \right\rfloor, \quad (16)$$

where $\lfloor \cdot \rfloor$ is the floor operation. The burst density as given in (16) effectively quantifies the supthreshold (threshold-exceeding) excitation. When the level set is not uniformly spaced, the I&FB neuron provides a nonuniform quantization of the supthreshold excitation. Representing a signal with sequences of pulses that correspond to a set of quantized approximations is commonly known as *pulse-code modulation*. When the quantized approximations are one-bit quantizations, it becomes PDM.

During $\Sigma\Delta$ modulation and Δ modulation, PDM signals naturally arise, representing finite-energy and band-limited waveforms via a number of pulses of short duration, appearing in proportion to the input signal amplitude. From a sample-and-encode perspective, PDM embeds a sample into the number of pulses fired near the time in which the sample was taken. Figure 7 illustrates $\Sigma\Delta$ and Δ modulators for PDM generation.

PDM might be considered a coarse-resolution PWM with a maximum of D possible pulses in each sampling interval T_n . Furthermore, the PPM signal that corresponds to a PDM signal rather than a PWM signal exhibits a discrete set of spiking instances, unlike the PWM-based PPM signal that can spike at any point in time. PDM connects to idealized bursting signals in (9) via convolution with the short pulse function $\rho(\cdot)$. Similar to the connection between PWM and spike trains, the sampling instances are not necessarily identical, yet using the finite variation of finite-energy band-limited signals that similar bounds on spiking and pulsing instances might be established.

Time-encoding machines with a variable refractory period in the burst mode, as discussed here, are qualitatively equivalent to PDM generating $\Sigma\Delta$ modulators within differentiation. The periodic resetting of the integrator in Figure 4 ensures that the time-encoding machine remains stable. In practice, however, due to better noise-shaping performance, a feedback loop illustrated in Figure 7 is preferred rather than resetting the integrator itself. Therefore, for the simple abstraction of the burst mode of integrate and fire-and-burst neurons discussed

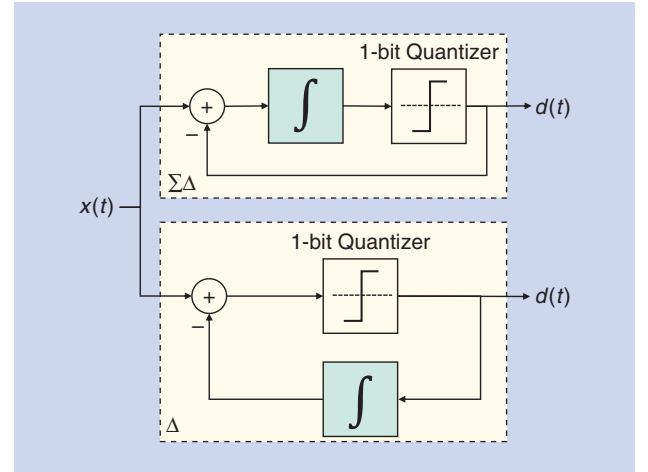


FIGURE 7. The PDM generation via $\Sigma\Delta$ and Δ modulators.

here, is conceptually equivalent to $\Sigma\Delta$ modulation (within differentiation) yet with inferior noise-shaping capabilities. Next, reconstruction from modulated signals is addressed.

Reconstruction from time-based representations

Time-based representations are not subject to saturations under dynamic range-exceeding excitations, yet such representations are not band-limited or of finite energy, even when the input signal is both. Unlike amplitude-based representations, time-based representations are not linear transformations and timing-to-sample information $\{\tau_n, w_n, d_n\} \rightarrow x_n$ is difficult to recover using analog circuits. However, it is generally feasible to use analog LPF from such representations to recover the original signal, provided sufficient oversampling was used at the generator. For idealized spike signals, analog LPF corresponds to sinc-interpolation from nonuniformly spaced samples [21].

In general, lossless representation conditions in the Landau sense assert only the existence of a reconstruction mechanism (see [15] and [21] and the references therein). However, finding such a reconstruction mechanism is generally nontrivial. Iterative algorithms employing problem-specific side information usually are computationally heavy. A widely accepted folk theorem states that, as long as Nyquist sampling rates hold on average, LPF reconstructions of the original signal

are possible from nonuniformly spaced samples. For instance, [21] investigates the mean-square error (MSE) of sinc interpolation from nonuniformly spaced samples at times $\{s_n\}$ of an arbitrary finite-energy band-limited signal $x(t)$.

$$\text{MSE} = \left\| x(t) - \sum_{n \in \mathbb{Z}} x(s_n) \text{sinc}(t - s_n) \right\|_2. \quad (17)$$

Such a reconstruction requires precise knowledge on nonuniform sampling instances and the corresponding samples $x(s_n)$ along with effective Nyquist sampling rate to hold [21].

However, spike trains and time-based representations embed the sampling information into the modulating waveform. Therefore, a form of decoding is necessary prior to any sample-based reconstruction mechanism. Decoding for such signal representations is in the form of analog information retrieval. For instance, measuring spiking instances ($s(t) \rightarrow \{\tau_n\}$) recovers the effective samples that generate a spike train. Estimating or counting the number of spikes in a burst (density) ($b(t) \rightarrow \{d_n\}$) recovers the sample encoded in a bursting signal. In the fairly more complex model of incorporating the spiking instances $\{\tau_{nm}\}_{m=1}^{d_n}$ contributing to the burst, the number of spikes as well as their spiking instances are needed ($b(t) \rightarrow \{d_n, \{\tau_{nm}\}_{m=1}^{d_n}\}$). Where pulse-based representations are concerned, measuring the duration or pulse-widths ($w(t) \rightarrow \omega_n$) for PWM effectively recovers the samples from the underlying signal $x(t)$. For PDM, counting the number of pulses ($d(t) \rightarrow d_n$) yields the sample information.

Such operations are simple logic and DSP operations, yet they are costly to implement in terms of circuit area and power consumption. For instance, TDCs that measure the pulsewidths of a PWM signal ($w(t) \rightarrow \omega_n$) are power-hungry, circuit-area-dominating units [22], [23]. Hence, for reconstructing the original signal $x(t)$ from its PWM counterpart, direct LPF ($w(t) \rightarrow \hat{x}(t)$) is employed as a low-circuit-complexity and power-efficient yet suboptimal reconstruction mechanism. Similar practices are also known for PDM microphones. Furthermore, biological nervous systems are continuous-time systems. Therefore, reconstruction strategies of form $\{s(t), w(t), b(t), d(t)\} \rightarrow x(t)$ are sought.

Figure 8 illustrates sinc interpolation-based reconstruction mechanisms from time-based representations. A time-decoding machine (TDM) has been proposed to reconstruct signals

from I&F generated spike trains [17]. The TDM uses values from the integrator of the I&F generator to scale the spikes and then uses analog LPF to reconstruct signal $\hat{x}(t)$. It is also known that when a PWM signal is generated by a natural oversampling modulator, low-pass reconstruction recovers the original signal asymptotically losslessly in the oversampling factor [15].

Even though both $s(t)$ and $w(t)$ represent the signal $x(t)$ losslessly in the Landau sense, it is still an open problem whether a pipeline $s(t) \rightarrow w(t) \rightarrow \tilde{x}(t)$, as shown in Figure 8, would recover the original signal in the MSE sense. Due to the spiking and pulsing instances being contained in bounded subintervals of the sampling intervals, the analysis in [15], [20], and [21] indicate that such a spike-to-pulse transformation prior to low-pass filtering might be viable. Furthermore, LPF of PDM signals is used in audio applications, such as those employing PDM microphones. The analysis in [15] extends to LPF reconstruction of finite-energy band-limited signals from their PDM counterparts by modeling PDM as a quantized PWM signal.

Stochastic models for time-based representations allow Bayesian models to be employed for information retrieval. The Poisson model in particular allows recursive methods of estimation for the rate function used as the modulating process (such as the optical signal in the optical sensing example) [26]–[28]. For spike trains that do not necessarily have Poisson statistics, continuous-time point processes have been employed [26]. Poisson statistics further allow techniques such as particle feedback filtering to become efficient for decoding spike trains [27]. The idea of LPF with prior oversampling manifests itself as a parameter-tuning problem for reconstruction from electromyographic signals [28].

Time-based representation of Poisson jump processes and finite-energy, band-limited, deterministic signals as well as the reconstruction of the modulating signals from their timing, density, or duration-modulated counterparts are of practical interest in many applications discussed in the “Introduction” section. Such applications are almost exclusively implemented through analog processing techniques. However, the digital transmission, storage, and processing of such time-domain signals are of further practical interest. Next, an efficient method for sampling time-modulated signals is addressed.

FRI sampling of time-based representations

When information from biological systems are incorporated into medical, science, or engineering applications, efficient sampling of spike-based representations becomes an important part of the application pipeline. Currently, in applications involving spike trains, massively high sampling rates are employed to reliably represent, store, and transmit biological signals, owing to their inherently large Fourier bandwidth. Emerging applications such as neural prosthetic devices require tens to hundreds of transmissions per task, yielding processing and transmission of high-rate signals that result in inefficiencies in terms of power consumption, hardware complexity, and communication bandwidth, among others. The finite rate of innovation (FRI) approach allows reliable representations of biological signals at low sampling rates (as compared to their Nyquist rates) [29], while allowing reliable reconstruction of the biological signals.

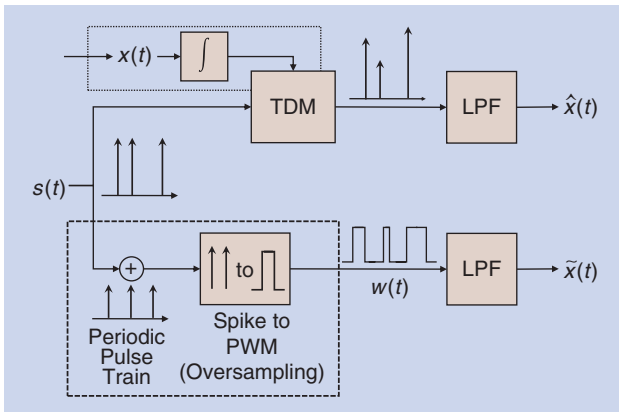


FIGURE 8. The LPF for reconstruction from spike trains and PWM.

FRI sampling asserts that a non-band-limited signal that lives in a finite-dimensional functional space (one that can be described within finite degrees of freedom per unit time) can be represented by a finite amount of samples projected on that space. As discussed, spiking time-based representations are not band-limited yet they are perfectly described by the set of spiking instances. For notational simplicity, consider a finite length spike train

$$s(t) = \sum_{n=0}^N \delta(t - \tau_n). \quad (18)$$

Here, the signal is of finite, known length $\tau_n \in [0, NT]$ for every n , where T is the length of sampling interval as defined in the “Signal Representation via Action Potentials and Time-Based Pulse Modulations” section.

Figure 9 illustrates a serially implemented FRI sampler that successively integrates the modulated signal, which results in functions

$$\begin{aligned} s_0(t) &= \int_0^t s(\tau) d\tau, \\ s_{l+1}(t) &= \int_0^t s_l(\tau) d\tau, \end{aligned} \quad (19)$$

with the corresponding FRI-based samples

$$y_l = s_l(NT) = \frac{1}{l!} \sum_{n=0}^{N-1} (NT - \tau_n)^l. \quad (20)$$

It is worth noting that the samples s_l can be acquired via an LPF bank, also known as the *sampling via LPF kernel*. Once the samples $\{s_l\}$ are acquired, they can be transmitted or stored at a much lower rate [29]. It is often of further interest to reconstruct the original biological waveform.

To reconstruct the spiking instances $\{\tau_n\}$ from the FRI samples $\{y_l\}$, we make a key observation: since samples in (20) are in the form of powersum series, hence $\{y_l\}$ uniquely determines $\{\tau_n\}$ using the roots of an annihilating filter of order $2N + 1$. Consider the matrix

$$S = \begin{bmatrix} y_N N! & \cdots & y_1 & y_0 \\ y_{N+1} (N+1)! & \cdots & y_2 2! & y_1 \\ \vdots & \cdots & \ddots & \vdots \\ y_{2N} (2N)! & \cdots & y_{N+1} (N+1)! & y_N N! \end{bmatrix}.$$

By the singular value decomposition, let $S = U\Sigma V^*$. Then the coefficients of the annihilating filter are given by the last column of V . The rest follows from solving for the roots $\{u_n\}$ of the annihilating filter. The spiking instances are given by $\tau_n = NT - u_n$ [29].

This sampling rule extends directly to PPM signals $p(t)$ that arise by convolving the spike train with a short pulse of length w . Let q_l be the FRI samples of $p(t)$. q_l then has the form

$$q_l = \frac{1}{l!} \sum_{n=0}^{N-1} (NT - \tau_n)^l - \frac{lNw^l}{l!}. \quad (21)$$

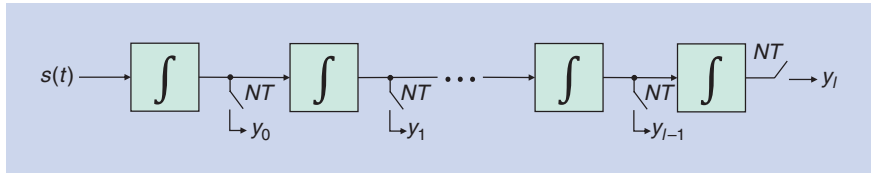


FIGURE 9. The serial FRI sampling of spike trains.

For $w \ll T$, the rule that recovers τ_n from the spike train $s(t)$ recovers τ_n from $p(t)$ as well. However, time-based representations such as PWM or PDM are commonly employed in purely analog pipelines. Interestingly, the FRI samples that correspond to a PWM signal have similar functional forms with those of spike trains and PPM signals. Formally, consider a PWM signal that comprises a finite number of pulses

$$w(t) = \sum_{n=0}^{N-1} u(t - nT) - u(t - nT - \omega_n).$$

Then, the sampler in Figure 9 produces samples z_l to represent the signal $w(t)$ as

$$z_l = \frac{1}{l!} \sum_{n=0}^{N-1} (\omega_n)^l + \frac{l}{l!} \sum_{n=0}^{N-1} (NT - \omega_n)^{l-1}.$$

Albeit not as straightforward, there exists a reconstruction mechanism for recovering ω_n from z_l in a manner similar to that used for spike trains [30]. FRI samples for PDM and burst modulation follow directly from the superposition of spikes/pulses. Consequently, time-based representations are characterized by similar FRI samples [see (20) and (21)] and can be transformed to one another efficiently using digital processing in addition to methods discussed in the “Signal Representation via Action Potentials and Time-Based Pulse Modulations” section.

Conclusions

Using time-based signal representations such as spikes and pulses, neurobiological systems achieve broad information-processing capabilities at power levels unattainable by many modern electronic systems. However, time-based representations have traditionally been used in several engineering applications ranging from optical sensing to power electronics and they are becoming prominent in emerging neuromorphic computing systems. In this article, we have focused on signal processing foundations for such time-based representations and explored the connections between action-potential neural responses and pulse-modulation techniques. Exciting engineering problems in implementing, modeling, and representing time-based pulsing-spiking systems will play an important role in the future of bioinspired circuits-and-systems design. We hope that this overview motivates further developments in the field of time-based signal processing.

Authors

Noyan C. Sevüktekin (sevukte2@illinois.edu) received his B.S. degree in electrical and electronics engineering from Bilkent University, Ankara, Turkey, in 2013. He received his M.S. degree in electrical and computer engineering from University

of Illinois at Urbana-Champaign in 2015, where he currently holds a research assistantship. His research interests include signal processing, statistical learning theory, and control theory. He received the best student paper award at the IEEE Asilomar Conference on Signals, Systems and Computers in 2017.

Lav R. Varshney (varshney@illinois.edu) received his B.S. degree in electrical and computer engineering from Cornell University, Ithaca, New York, in 2004 and his S.M., E.E., and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 2006, 2008, and 2010, respectively. He is an assistant professor of electrical and computer engineering at the University of Illinois at Urbana-Champaign. His research interests include neuroscience, neuroengineering, signal processing, information theory, and artificial intelligence. His work appears in the anthology *The Best Writing on Mathematics 2014* (Princeton University Press), and he was selected to present at the 2017 World Science Festival. He appears on the List of Teachers Ranked as Excellent and has been named a Center for Advanced Study Fellow at the University of Illinois. He serves on the advisory board of the IBM Watson AI XPRIZE.

Pavan K. Hanumolu (hanumolu@illinois.edu) received his B.E. degree in electrical engineering from the Birla Institute of Technology and Science, Pilani, India, in 1998 and his Ph.D. degree from the School of Electrical Engineering and Computer Science at Oregon State University, Corvallis, in 2006, where he subsequently served as a faculty member until 2013. He is currently a professor in the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign. His research interests include energy-efficient integrated circuit implementation of analog and digital signal processing, sensor interfaces, wireline communication systems, and power conversion.

Andrew C. Singer (acsinger@illinois.edu) received his S.B., S.M., and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge. Since 1998, he has been on the faculty of the Electrical and Computer Engineering Department, University of Illinois at Urbana-Champaign, where he currently holds a Fox Family Endowed Professorship and serves as associate dean for innovation and entrepreneurship in the College of Engineering. He was a cofounder of Intersymbol Communications and OceanComm and received the IEEE Journal of Solid-State Circuits Best Paper Award and the IEEE Signal Processing Magazine Award. He is a Fellow of the IEEE and, in 2014, was named a Distinguished Lecturer of the IEEE Signal Processing Society.

References

- [1] S. R. Lockery and M. B. Goodman, "The quest for action potentials in *C. elegans* neurons hits a plateau," *Nat. Neurosci.*, vol. 12, no. 4, pp. 377–378, 2009. doi: 10.1038/nn0409-377.
- [2] J. Hsu, "How IBM got brainlike efficiency from the trueth chip," *IEEE Spectr.*, vol. 51, no. 10, pp. 17–19, 2014. doi: 10.1109/MSPEC.2014.6905473.
- [3] S. Suh, "Pulse width modulation for analog fiber-optic communications," *Lightwave Technol. J.*, vol. 5, no. 1, pp. 102–112, 1987. doi: 10.1109/JLT.1987.1075401.
- [4] M. Mansuripur, *The Physical Principles of Magneto-Optical Recording*. Cambridge, U.K.: Cambridge Univ. Press, 1998.

- [5] C. Pascual, Z. Song, P. Krein, D. Sarwate, P. Midya, and W. Roeckner, "High-fidelity PWM inverter for digital audio amplification: Spectral analysis, real-time DSP implementation, and results," *IEEE Trans. Power Electron.*, vol. 18, no. 1, pp. 473–485, Jan 2003. doi: 10.1109/TPEL.2002.807102.
- [6] S. Rao, B. Young, A. Elshazly, W. Yin, N. Sasidhar, and P. K. Hanumolu, "A 71db SFDR open loop VCO-based ADC using 2-level PWM modulation," in *Proc. IEEE Symp. VLSI Circuits*, 2011, pp. 270–271. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/5986448>
- [7] D. G. Holmes and T. A. Lipo, *Pulse Width Modulation for Power Converters: Principles and Practice*. Hoboken, NJ: Wiley, 2003.
- [8] T. Kite, "Understanding PDM digital audio," Audio Precision, 2012. [Online]. Available: http://users.ece.utexas.edu/~bevans/courses/rtdsp/lectures/10_Data_Conversion/AP_Understanding_PDM_Digital_Audio.pdf
- [9] J. J. Hopfield, "Pattern recognition computation using action potential timing for stimulus representation," *Nature*, vol. 376, pp. 33–36, 1995. doi: 10.1038/376033a0.
- [10] E. Cotterill, P. Charlesworth, C. W. Thomas, O. Paulsen, and S. J. Eglen, "A comparison of computational methods for detecting bursts in neuronal spike trains and their application to human stem cell-derived neuronal networks," *J. Neurophys.*, vol. 116, no. 2, pp. 306–321, 2016. doi: 10.1152/jn.00093.2016.
- [11] B. P. Bean, "The action potential in mammalian central neurons," *Nature Rev. Neurosci.*, vol. 8, no. 6, pp. 451–465, 2007. doi: 10.1038/nrn2148.
- [12] S. P. Strong, R. Koberle, R. R. d. R. van Steveninck, and W. Bialek, "Entropy and information in neural spike trains," *Phys. Rev. Lett.*, vol. 80, no. 1, p. 197, 1998. doi: 10.1103/PhysRevLett.80.197.
- [13] S. B. Laughlin and T. J. Sejnowski, "Communication in neuronal networks," *Sci.*, vol. 301, no. 5641, pp. 1870–1874, Sept. 2003. doi: 10.1126/science.1089662.
- [14] Z. Song and D. V. Sarwate, "The frequency spectrum of pulse width modulated signals," *Signal Process.*, vol. 83, no. 10, pp. 2227–2258, 2003. doi: 10.1016/S0165-1684(03)00164-6.
- [15] N. C. Seviktekin and A. C. Singer, "Lossless natural sampling for PWM generation," in *IEEE 51st Asilomar Conf. Signals, Systems, and Computers*, 2017, pp. 1935–1939. doi: 10.1109/ACSSC.2017.8335702.
- [16] A. Papoulis, *Signal Analysis*. New York: McGraw-Hill, 1978.
- [17] A. A. Lazar, "Time encoding with an integrate-and-fire neuron with a refractory period," *Neurocomput.*, vol. 58–60, pp. 53–58, 2004. doi: 10.1016/S0925-2312(04)00017-7.
- [18] E. M. Izhikevich, "Which model to use for cortical spiking neurons?" *IEEE Trans. Neural Netw.*, vol. 15, no. 5, pp. 1063–1070, 2004. doi: 10.1109/TNN.2004.832719.
- [19] A. A. Lazar, "A simple model of spike processing," *Neurocomput.*, vol. 69, no. 10–12, pp. 1081–1085, 2006. doi: 10.1016/j.neucom.2005.12.050.
- [20] A. A. Lazar and L. T. Tóth, "Perfect recovery and sensitivity analysis of time encoded bandlimited signals," *IEEE Trans. Circuits Syst. I*, vol. 51, no. 10, pp. 2060–2073, 2004. doi: 10.1109/TCSI.2004.835026.
- [21] S. Maymon and A. V. Oppenheim, "Sinc interpolation of nonuniform samples," *IEEE Trans. Signal Process.*, vol. 59, no. 10, pp. 4745–4758, Oct 2011. doi: 10.1109/TSP.2011.2160054.
- [22] V. Dhanasekaran, M. Gambhir, M. M. Elsayed, E. Sanchez-Sinencio, J. Silva-Martinez, C. Mishra, L. Chen, and E. Pankratz, "A 20mhz BW 68db DR CT $\Delta\Sigma$ ADC based on a multi-bit time-domain quantizer and feedback element," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2009, pp. 174–175. doi: 10.1109/ISSCC.2009.4977364.
- [23] S. Naraghi, M. Courcy, and M. P. Flynn, "A 9b 14 μ w 0.06mm2 ppm ADC in 90nm digital CMOS," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb 2009, pp. 168–169.
- [24] K. Yao and J. Thomas, "On some stability and interpolatory properties of non-uniform sampling expansions," *IEEE Trans. Circuit Theory*, vol. 14, no. 4, pp. 404–408, Dec. 1967. doi: 10.1109/TCT.1967.1082745.
- [25] N. C. Seviktekin and A. C. Singer, "A performance bound on low-pass reconstruction from PWM signals," in *Proc. IEEE Int. Conf. Communications*, 2015, pp. 4931–4936. doi: 10.1109/ICC.2015.7249104.
- [26] O. Bobrowski, R. Meir, and Y. C. Eldar, "Bayesian filtering in spiking neural networks: Noise, adaptation, and multisensory integration," *Neural Comput.*, vol. 21, no. 5, pp. 1277–1320, 2009. doi: 10.1162/neco.2008.01-08-692.
- [27] A. E. Brockwell, A. L. Rojas, and R. Kass, "Recursive Bayesian decoding of motor cortical signals by particle filtering," *J. Neurophys.*, vol. 91, no. 4, pp. 1899–1907, 2004. doi: 10.1152/jn.00438.2003.
- [28] T. D. Sanger, "Bayesian filtering of myoelectric signals," *J. Neurophys.*, vol. 97, no. 2, pp. 1839–1845, 2007. doi: 10.1152/jn.00936.2006.
- [29] B. D. He, A. Wein, L. R. Varshney, J. Kusuma, A. G. Richardson, and L. Srinivasan, "Generalized analog thresholding for spike acquisition at ultralow sampling rates," *J. Neurophys.*, vol. 114, no. 1, pp. 746–760, 2015. doi: 10.1152/jn.00623.2014.
- [30] M. Vetterli, J. Kovačević, and V. Goyal, *Foundations of Signal Processing*. Cambridge, U.K.: Cambridge Univ. Press, 2014.

Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke

Surrogate Gradient Learning in Spiking Neural Networks

*Bringing the power of gradient-based optimization
to spiking neural networks*



©ISTOCKPHOTO.COM/JUST_SUPER

Spiking neural networks (SNNs) are nature's versatile solution to fault-tolerant, energy-efficient signal processing. To translate these benefits into hardware, a growing number of neuromorphic spiking NN processors have attempted to emulate biological NNs. These developments have created an imminent need for methods and tools that enable such systems to solve real-world signal processing problems. Like conventional NNs, SNNs can be trained on real, domain-specific data; however, their training requires the overcoming of a number of challenges linked to their binary and dynamical nature. This article elucidates step-by-step the problems typically encountered when training SNNs and guides the reader through the key concepts of synaptic plasticity and data-driven learning in the spiking setting. Accordingly, it gives an overview of existing approaches and provides an introduction to surrogate gradient (SG) methods, specifically, as a particularly flexible and efficient method to overcome the aforementioned challenges.

Introduction

Biological SNNs are a highly efficient solution to the problem of signal processing. Therefore, taking inspiration from the brain is a natural approach to engineering more efficient computing architectures. In the area of machine learning, recurrent NNs (RNNs), a class of stateful NNs whose internal state evolves with time (see "Recurrent Neural Networks"), have proven highly effective at solving real-time pattern recognition and noisy time-series prediction problems [1]. RNNs and biological NNs share several properties, such as a similar general architecture, temporal dynamics, and learning through weight adjustments. Based on these similarities, a growing body of work is now establishing formal equivalences between RNNs and networks of spiking leaky integrate-and-fire (LIF) neurons, which are widely used in computational neuroscience [2]–[5].

RNNs are typically trained using an optimization procedure in which the parameters or weights are adjusted to minimize a given objective function. Efficiently training large-scale RNNs is challenging due to a variety of extrinsic factors, such as noise and nonstationarity of the data, but also due to the inherent difficulties

of optimizing functions with long-range temporal and spatial dependencies. In SNNs and binary RNNs, these difficulties are compounded by the nondifferentiable dynamics implied by the binary nature of their outputs. Although a considerable body of work has successfully demonstrated the training of two-layer SNNs [6]–[8] without hidden units as well as networks with recurrent synaptic connections [9], [10], the ability to train deeper SNNs with hidden layers has remained a major obstacle. Because hidden units and depth are crucial for efficiently solving many real-world problems, overcoming this obstacle is vital.

Recurrent Neural Networks

Recurrent neural networks (RNNs) are networks of interconnected units, i.e., neurons, in which their network state at any point in time is a function of both external input and the network's state at the previous time point, as shown in Figure S1. More precisely, the dynamics of a network with L layers is given by:

$$\begin{aligned} \mathbf{y}^{(l)}[n] &= \sigma(\mathbf{a}^{(l)}[n]) \quad \text{for } l = 1, \dots, L, \\ \mathbf{a}^{(l)}[n] &= \mathbf{V}^{(l)} \mathbf{y}^{(l)}[n-1] + \mathbf{W}^{(l)} \mathbf{y}^{(l-1)}[n-1] \quad \text{for } l = 1, \dots, L, \\ \mathbf{y}^{(0)}[n] &\equiv \mathbf{x}[n], \end{aligned}$$

where $\mathbf{a}^{(l)}[n]$ is the state vector of the neurons at layer l , σ is an activation function, and $\mathbf{V}^{(l)}$ and $\mathbf{W}^{(l)}$ are the recurrent and feedforward weight matrices of layer l , respectively. External inputs $\mathbf{x}[n]$ typically arrive at the first layer. Nonscalar quantities are typeset in boldface.

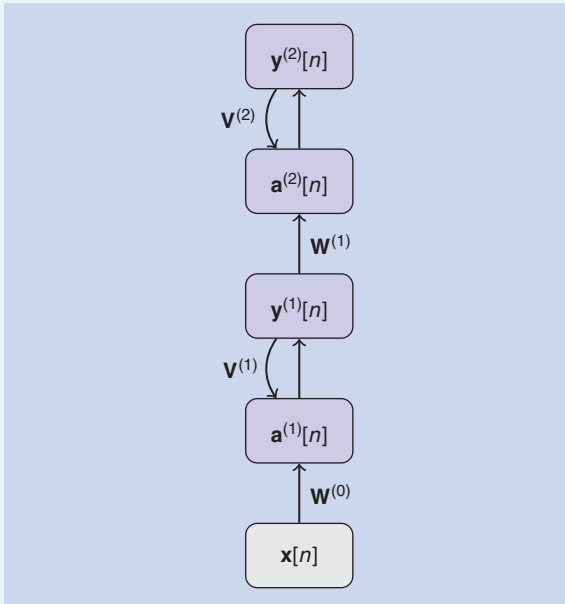


FIGURE S1. One popular RNN structure arranges neurons in multiple layers, where every layer is recurrently connected and also receives input from the previous layer.

As network models grow larger and make their way into embedded and automotive applications, their power efficiency becomes increasingly important. Simplified NN architectures that can run natively and efficiently on dedicated hardware are now being devised. This includes, e.g., networks of binary neurons or neuromorphic hardware that emulate the dynamics of SNNs [11]. Both types of networks dispense with energetically costly floating-point multiplications, making them particularly advantageous for low-power applications compared to NNs executed on conventional hardware.

These new hardware developments have created an imminent need for tools and strategies that enable efficient inference and learning in SNNs and binary RNNs. In this article, we discuss and address the inherent difficulties in training SNNs with hidden layers and introduce various strategies and approximations used to successfully implement them. (A repository containing tutorials for SG learning in SNNs can be found at: <https://github.com/surrogate-gradient-learning>.)

Understanding SNNs as RNNs

We begin by formally mapping SNNs to RNNs. Formulating SNNs as RNNs will allow us to directly transfer and apply existing training methods for RNNs and will serve as the conceptual framework for the rest of this article.

Before we proceed, we must make a note about terminology. We use the term *RNNs* in its widest sense to refer to networks whose state evolves in time according to a set of recurrent dynamical equations. Such dynamical recurrence can be because of the explicit presence of recurrent synaptic connections between neurons in the network. This is the common understanding of what an RNN is. But, importantly, dynamical recurrence can also arise in the absence of recurrent connections. This happens, e.g., when stateful neuron or synapse models, which have internal dynamics, are used. Because the network's state at a particular time step recurrently depends on its state in previous time steps, these dynamics are intrinsically recurrent. In this article, we use the term *RNN* for networks exhibiting either or both types of recurrence. Moreover, we introduce the term *recurrently connected NN (RCNN)* for the subset of networks that have explicit recurrent synaptic connections. We now describe the mathematical treatment of RCNNs, which closely resembles that of RNNs.

We first introduce an LIF neuron model with current-based synapses, which has wide use in computational neuroscience [12]. Next, we reformulate this model in discrete time and show its formal equivalence to an RNN with binary activation functions. Readers familiar with an LIF neuron model can skip the steps in “Recurrent Neural Networks” as well as (1)–(4), up to (5).

An LIF neuron in layer l with index i can formally be described in differential form as

$$\tau_{\text{mem}} \frac{dU_i^{(l)}}{dt} = -(U_i^{(l)} - U_{\text{rest}}) + RI_i^{(l)}, \quad (1)$$

where $U_i^{(l)}(t)$ is the membrane potential, U_{rest} is the resting potential, τ_{mem} is the membrane time constant, R is the input resistance, and $I_i(t)$ is the input current [12]. Equation

(1) shows that $U_i^{(l)}$ acts as a leaky integrator of the input current $I_i^{(l)}$. Neurons emit spikes to communicate their output to other neurons when their membrane voltage reaches the firing threshold ϑ . After each spike, the membrane voltage $U_i^{(l)}$ is reset to the resting potential U_{rest} (Figure 1). Due to this reset, (1) describes only the subthreshold dynamics of an LIF neuron, i.e., the dynamics in absence of spiking output of the neuron.

In SNNs, the input current is typically generated by synaptic currents triggered by the arrival of presynaptic spikes $S_j^{(l)}(t)$. When working with differential equations, it is convenient to denote a spike train $S_j^{(l)}(t)$ as a sum of Dirac delta functions $S_j^{(l)}(t) = \sum_{s \in C_j^{(l)}} \delta(t - s)$, where s runs over the firing times $C_j^{(l)}$ of neuron j in layer l .

Synaptic currents follow specific temporal dynamics themselves. A common first-order approximation is to model their time course as an exponentially decaying current following each presynaptic spike. Moreover, we assume that synaptic currents sum linearly. The dynamics of these operations are given by

$$\frac{dI_i^{(l)}}{dt} = -\underbrace{\frac{I_i^{(l)}(t)}{\tau_{\text{syn}}}}_{\text{exp. decay}} + \underbrace{\sum_j W_{ij}^{(l)} S_j^{(l-1)}(t)}_{\text{feedforward}} + \underbrace{\sum_j V_{ij}^{(l)} S_j^{(l)}(t)}_{\text{recurrent}}, \quad (2)$$

where the sum runs over all presynaptic neurons j and $W_{ij}^{(l)}$ are the corresponding afferent weights from the layer below. Further, the $V_{ij}^{(l)}$ corresponds to explicit recurrent connections within each layer. Because of this property, we can simulate a single LIF neuron with two linear differential equations whose initial conditions change instantaneously whenever a spike occurs. Through this property, we can incorporate the reset term in (1) through an extra term that instantaneously decreases the membrane potential by the amount $(\vartheta - U_{\text{rest}})$ whenever the neuron emits a spike:

$$\frac{dU_i^{(l)}}{dt} = -\frac{1}{\tau_{\text{mem}}}((U_i^{(l)} - U_{\text{rest}}) + RI_i^{(l)}) + S_i^{(l)}(t)(U_{\text{rest}} - \vartheta). \quad (3)$$

It is customary to approximate the solutions of (2) and (3) numerically in discrete time and to express the output spike train $S_i^{(l)}[n]$ of neuron i in layer l at time step n as a nonlinear function of the membrane voltage $S_i^{(l)}[n] \equiv \Theta(U_i^{(l)}[n] - \vartheta)$, where Θ denotes the Heaviside step function and ϑ corresponds to the firing threshold. Without loss of generality, we set $U_{\text{rest}} = 0$, $R = 1$, and $\vartheta = 1$. When using a small simulation time step $\Delta t > 0$, (2) is well approximated by

$$I_i^{(l)}[n+1] = \alpha I_i^{(l)}[n] + \sum_j W_{ij}^{(l)} S_j^{(l-1)}[n] + \sum_j V_{ij}^{(l)} S_j^{(l)}[n], \quad (4)$$

with the decay strength $\alpha \equiv \exp(-(\Delta t/\tau_{\text{syn}}))$. Note that $0 < \alpha < 1$ for finite and positive τ_{syn} . Moreover, $S_j^{(l)}[n] \in \{0, 1\}$. We use n to denote the time step to emphasize the discrete dynamics. We can now express (3) as

$$U_i^{(l)}[n+1] = \beta U_i^{(l)}[n] + I_i^{(l)}[n] - S_i^{(l)}[n] \quad (5)$$

with $\beta \equiv \exp(-(\Delta t/\tau_{\text{mem}}))$.

Equations (4) and (5) characterize the dynamics of an RNN. Specifically, the state of neuron i is given by the instantaneous

synaptic currents $I_i^{(l)}$ and the membrane voltage $U_i^{(l)}$ (see “Recurrent Neural Networks”). The computations necessary to update the cell state can be unrolled in time, as is best illustrated by the computational graph shown in Figure 2.

We have now seen that SNNs constitute a special case of RNNs; however, we have not yet explained how their parameters are set to implement a specific computational function. This is the focus of the rest of this article, in which we present a variety of learning algorithms that systematically change the parameters toward implementing specific functionalities.

Methods for training RNNs

Powerful machine-learning methods are able to train RNNs for a variety of tasks ranging from time-series prediction, to language translation, to automatic speech recognition [1]. In this section, we discuss the most common methods before analyzing their applicability to SNNs.

There are several common ingredients that define the training process in RNNs. The first ingredient is a cost or loss function, which is minimized when the network’s response corresponds to the desired behavior. In time-series prediction, e.g., this loss could be the squared difference between the predicted and true values. The second ingredient is a mechanism that updates the network’s weights to minimize the loss. One of the simplest and most powerful mechanisms used to achieve this is to perform gradient descent on the loss function. In network architectures with hidden units (i.e., units whose activity affect the loss indirectly through other units), the parameter updates contain terms that relate to the activity and weights of the downstream units they project to. Gradient-descent learning solves this credit assignment problem by providing explicit expressions for these updates through the chain rule of derivatives.

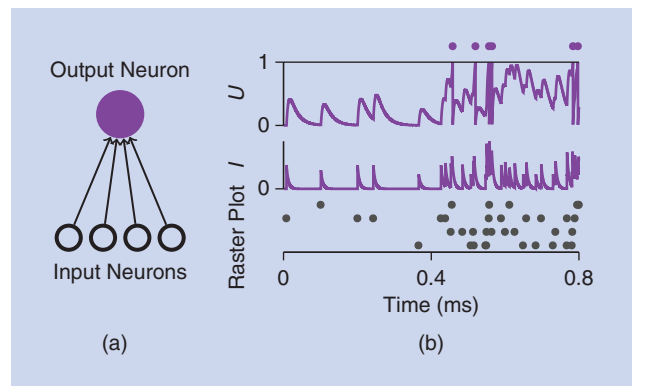


FIGURE 1. An example of LIF neuron dynamics. (a) The schematic of a network setup. Four input neurons connect to one postsynaptic neuron. (b) The input and output activity over time. At the bottom of (b) is the Raster plot, which shows activity of the four input neurons; in the middle is the synaptic current I ; and at the top is the membrane potential U of the output neuron as a function of time, with output spikes shown as points. During the first 0.4 s, the dynamics are strictly “subthreshold,” and individual postsynaptic potentials are clearly discernible. Only when multiple postsynaptic potentials start to sum up is the neuronal firing threshold (dashed) reached and output spikes generated.

As we will now see, the learning of hidden-unit parameters depends on an efficient method to compute these gradients. When discussing these methods, we distinguish between solving the spatial credit assignment problem, which affects multilayer perceptrons (MLPs) and RNNs in the same way, and the temporal credit assignment problem, which only occurs in RNNs. In the following section, we discuss the common algorithms that provide both types of credit assignment.

Spatial credit assignment

To train MLPs, credit or blame needs to be assigned spatially across the layers and their respective units. This spatial credit assignment problem is solved most commonly by the backpropagation (BP)-of-error algorithm (see “The Gradient Backpropagation Rule for Neural Networks”). In its simplest form, this algorithm propagates errors “backward” from the output of the network to upstream neurons. Using BP to adjust hidden-layer weights ensures that the weight update will reduce the cost function for the current training example, provided the learning rate is small enough. Although this theoretical guarantee is desirable, it comes at the cost of certain communication requirements, i.e., that gradients must be communicated back through the network, and increased memory requirements as the neuron states must be kept in memory until the errors become available.

Temporal credit assignment

When training RNNs, we also must consider the temporal interdependencies of network activity. This requires solving the

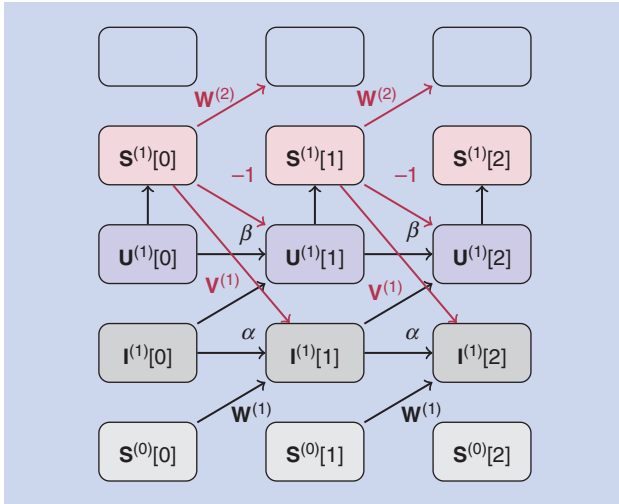


FIGURE 2. An illustration of the computational graph of an SNN in discrete time. The time steps flow from left to right. Input spikes $\mathbf{S}^{(0)}$ are fed into the network from the bottom and propagate upward to higher layers. The synaptic currents $\mathbf{I}^{(1)}$ are decayed by α in each time step and fed into the membrane potentials $\mathbf{U}^{(1)}$. The $\mathbf{U}^{(1)}$ are similarly decaying over time, as characterized by β . Spike trains $\mathbf{S}^{(1)}$ are generated by applying a threshold nonlinearity to the membrane potentials $\mathbf{U}^{(1)}$ in each time step. Spikes causally affect the network state (red connections). First, each spike causes the membrane potential of the neuron that emits the spike to be reset. Second, each spike may be communicated to the same neuronal population via recurrent connections $\mathbf{V}^{(1)}$. Finally, it may also be communicated via $\mathbf{W}^{(2)}$ to another downstream network layer or, alternatively, a readout layer on which a cost function is defined.

temporal credit assignment problem shown in Figure 2. There are two common methods used to achieve this:

- 1) *The “backward” method:* This method applies the same strategies used for spatial credit assignment by “unrolling” the network in time (see “The Gradient Backpropagation Rule for Neural Networks”). BP through time (BPTT) solves the temporal credit assignment problem by backpropagating errors through the unrolled network. This method works backward through time after completing a forward pass. The use of standard BP on the unrolled network directly enables the use of autodifferentiation tools offered in modern machine-learning toolkits [3], [13].
- 2) *The forward method:* In some situations, it is beneficial to propagate all necessary information for gradient computation forward in time [14]. This formulation is achieved by computing the gradient of a cost function $\mathcal{L}[n]$ and maintaining the recursive structure of the RNN. For example, the “forward gradient” of the feedforward weight \mathbf{W} becomes

$$\Delta W_{ij}^{[m]} \propto \frac{\partial \mathcal{L}[n]}{\partial W_{ij}^{[m]}} = \sum_k \frac{\partial \mathcal{L}[n]}{\partial y_k^{[L]}} P_{ijk}^{[L,m]}[n],$$

with

$$\begin{aligned} P_{ijk}^{(l,m)}[n] &= \frac{\partial}{\partial W_{ij}^{(l,m)}} y_k^{(l)}[n] \\ P_{ijk}^{(l,m)}[n] &= \sigma'(a_k^{(l)}[n]) \left(\sum_j V_{ij'}^{(l)} P_{ij'k}^{(l,m)}[n-1] \right. \\ &\quad \left. + \sum_j W_{ij'}^{(l)} P_{ij'k}^{(l-1,m)}[n-1] + \delta_{im} y_i^{(l-1)}[n-1] \right). \end{aligned} \quad (6)$$

Gradients, with respect to recurrent weights $V_{ij}^{(l)}$, can be computed in a similar fashion [14].

The backward optimization method is generally more efficient in terms of computation, but requires the maintaining of all inputs and activations for each time step. Thus, its space complexity for each layer is $O(NT)$, where N is the number of neurons per layer, and T is the number of time steps. Conversely, the forward method requires maintaining variables $P_{ijk}^{(l,m)}$, resulting in an $O(N^3)$ space complexity per layer. Although $O(N^3)$ is not a favorable scaling compared to $O(NT)$ for large N , simplifications of the computational graph can reduce the memory complexity of the forward method to $O(N^2)$ [2], [15], or even $O(N)$ [4]. These simplifications also reduce computational complexity, rendering the scaling of forward algorithms comparable to, or better than, BPTT. Such simplifications are at the core of several successful approaches, which we describe in the “Applications” section. Furthermore, the forward method is more appealing from a biological point of view, since the learning rule can be made consistent with synaptic plasticity in the brain and “three-factor” rules, as discussed in the “Supervised Learning With Local Three-Factor Learning Rules” section. In summary, efficient algorithms used to train RNNs exist. In the following section, we focus on training SNNs.

Credit assignment with spiking neurons: Challenges and solutions

Thus far, we have discussed common algorithmic solutions used for training RNNs. Before these solutions can be applied

to SNNs, however, two key challenges need to be overcome. The first challenge concerns the nondifferentiability of the spiking nonlinearity. Equations (S2) and (6) reveal that the expressions for both the forward- and backward-learning methods contain the derivative of the neural activation function $\sigma' \equiv \partial y_i^{(l)} / \partial a_i^{(l)}$ as a multiplicative factor. For a spiking neuron, however, we have $S(U(t)) = \Theta(U(t) - \vartheta)$, whose derivative is zero everywhere except at $U = \vartheta$, where it is ill defined (see Figure 3). This all-or-nothing behavior of the binary spiking nonlinearity stops gradients from “flowing” and makes LIF neurons unsuitable for gradient-based optimization. The same issue occurs in binary neurons, and some of the solutions proposed in this section are inspired by methods first developed in binary networks [16], [17].

The second challenge concerns the implementation of the optimization algorithm itself. Standard BP can be expensive in terms of computation, memory, and communication and may be poorly suited to the constraints dictated by the hardware that implements it (e.g., a computer, brain, or neuromorphic device). Processing in dedicated neuromorphic hardware and, more gen-

erally, non-von Neumann computers may have specific locality requirements (see “Local Models of Computation”), which can complicate matters. On such hardware, the forward approach may therefore be preferable. In practice, however, the scaling of both methods ($O(N^3)$ and $O(NT)$) has proven unsuitable for many SNN models. For example, the size of the convolutional SNN models trained with BPTT for gesture classification [20] are graphics processing unit (GPU) memory bounded. Additional simplifying approximations that reduce the complexity of the forward method will be discussed in greater detail. In the following sections, we describe approximate solutions to these challenges that make learning in SNNs more tractable.

To overcome the first challenge in training SNNs, which is concerned with the discontinuous spiking nonlinearity, several approaches have been devised with varying degrees of success. The most common approaches can be coarsely classified into the following categories: 1) resorting to entirely biologically inspired local learning rules for the hidden units; 2) translating conventionally trained “rate-based” NNs to SNNs; 3) smoothing the network model to be continuously differentiable; or

The Gradient Backpropagation Rule for Neural Networks

The task of learning is to minimize a cost function \mathcal{L} over the entire data set. In a neural network (NN), this can be achieved by gradient descent, which modifies the network parameters \mathbf{W} in the direction opposite to the gradient

$$\mathbf{W}_{ij} \leftarrow \mathbf{W}_{ij} - \eta \Delta \mathbf{W}_{ij}, \text{ where } \Delta \mathbf{W}_{ij} = \frac{\partial \mathcal{L}}{\partial \mathbf{W}_{ij}} = \frac{\partial \mathcal{L}}{\partial y_i} \frac{\partial y_i}{\partial a_i} \frac{\partial a_i}{\partial \mathbf{W}_{ij}},$$

with $a_i = \sum_j \mathbf{W}_{ij} x_j$ as the total input to the neuron, y_i as the output of neuron i , and η as a small learning rate. The first term is the error of neuron i , and the second term reflects the sensitivity of the neuron output to changes in the parameter. In multilayer networks, gradient descent is expressed as the backpropagation (BP) of the errors starting from the prediction (output) layer to the inputs. Using superscripts $l = 0, \dots, L$ to denote the layer (0 is input, L is output)

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{ij}^{(l)}} \mathcal{L} = \delta_i^{(l)} y_j^{(l-1)}, \text{ where } \delta_i^{(l)} = \sigma'(a_i^{(l)}) \sum_k \delta_k^{(l+1)} \mathbf{W}_{ik}^{\top, (l)}, \quad (\text{S1})$$

where σ' is the derivative of the activation function, $\delta_i^{(l)} = \partial \mathcal{L} / \partial y_i^{(l)}$ is the error of output neuron i , and $y_i^{(0)} = x_i$ and \top indicate the transpose.

This update rule is ubiquitous in deep learning and known as the *gradient BP algorithm* [1]. Learning is typically carried out in forward passes (evaluation of the NN activities) and backward passes (evaluation of δ s).

The same rule can be applied to recurrent NNs. In this case, the recurrence is “unrolled,” meaning that an auxiliary network is created by making copies of the network for

each time step, as depicted in Figure S2. The unrolled network is simply a deep network with shared feedforward weights $\mathbf{W}^{(l)}$ and recurrent weights $\mathbf{V}^{(l)}$, on which the standard BP applies

$$\begin{aligned} \Delta \mathbf{W}_{ij}^{(l)} &\propto \frac{\partial}{\partial \mathbf{W}_{ij}^{(l)}} \mathcal{L}[n] = \sum_{m=0}^n \delta_i^{(l)}[m] y_j^{(l-1)}[m], \text{ and} \\ \Delta \mathbf{V}_{ij}^{(l)} &\propto \frac{\partial}{\partial \mathbf{V}_{ij}^{(l)}} \mathcal{L}[n] = \sum_{m=1}^n \delta_i^{(l)}[m] y_j^{(l)}[m-1] \\ \delta_i^{(l)}[m] &= \sigma'(a_i^{(l)}[m]) \left(\sum_k \delta_k^{(l+1)}[m] \mathbf{W}_{ik}^{\top, (l)} + \sum_k \delta_k^{(l)}[m+1] \mathbf{V}_{ik}^{\top, (l)} \right). \end{aligned} \quad (\text{S2})$$

Applying BP to an unrolled network is referred to as *BP through time*.

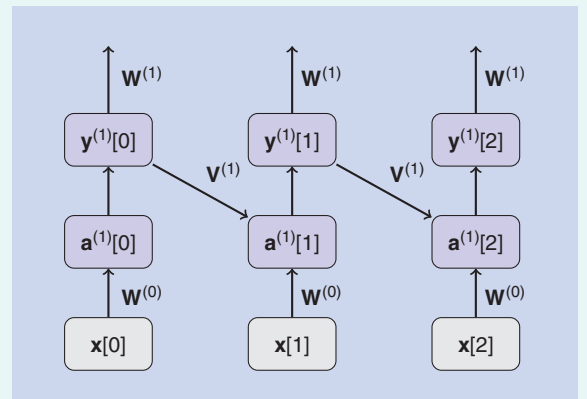


FIGURE S2. An “unrolled” recurrent NN.

4) defining an SG as a continuous relaxation of the real gradients. Approaches pertaining to biologically motivated local learning rules (i.e., category 1) and network translation (i.e., category 2) have been reviewed extensively [5], [21]. In this article, we therefore focus on the latter two supervised approaches (i.e., categories 3 and 4), which we will refer to as the *smoothed* and *SG approaches*, respectively. First, we review existing literature on common “smoothing” approaches before turning to an in-depth discussion of how to build functional SNNs using SG methods.

Smoothed SNNs

The defining characteristic of smoothed SNNs is that their formulation ensures well-behaved gradients, which are directly suitable for optimization. Smooth models can be further categorized into 1) soft nonlinearity models; 2) probabilistic models, for which gradients are well defined only in expectation, or models that either rely entirely on 3) rate; or (4) single-spike temporal codes.

Gradients in soft nonlinearity models

This approach can, in principle, be applied directly to all spiking neuron models, which explicitly include a smooth spike-generating process. This includes, e.g., the Hodgkin–Huxley, Morris–Lecar, and FitzHugh–Nagumo models [12]. In practice, this approach has been applied successfully only by Huh and Sejnowski [22], using an augmented IF model in which the binary spiking nonlinearity was replaced by a continuous-valued gating function. The resulting network constitutes an RCNN, which can be optimized using standard methods of BPTT or real-time recurrent learning (RTRL). Importantly, the soft threshold models compromise on one of the key features of SNNs, i.e., the binary spike propagation.

Gradients in probabilistic models

Another example of smooth models is binary probabilistic models. In simple terms, stochasticity effectively smooths out discontinuous binary nonlinearity, which makes it possible to define a gradient on expectation values. Binary probabilistic

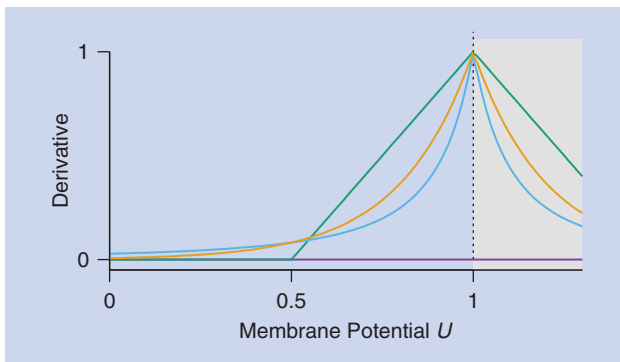


FIGURE 3. Commonly used surrogate derivatives. The step function has a zero derivative (violet) everywhere except at 0, where it is ill defined. The green (piecewise linear) [3], [18], [19], blue (derivative of a fast sigmoid) [2], and yellow (exponential) [13] lines are examples of surrogate derivatives that have been used to train SNNs. The gray shaded area is inaccessible due to the spike generation. Note that we have plotted absolute values and rescaled the axes on a per-function-basis for illustration purposes.

models have been objects of extensive study in machine-learning literature, mainly in the context of (restricted) Boltzmann machines [23]. Similarly, the propagation of gradients has been studied for binary stochastic models [17]. Probabilistic models are practically useful because the log-likelihood of a spike train is a smooth quantity, which can be optimized using gradient descent [24]. Although this insight was first discovered in networks without hidden units, the same ideas were later extended to multilayer networks [25]. Similarly, Guerguiev et al. [26] used probabilistic neurons to study biologically plausible ways of propagating error or target signals using segregated dendrites (see the “Feedback Alignment and Random Error BP” section). In a similar vein, variational learning approaches were shown to be capable of learning useful hidden-layer representations in SNNs [27]–[29]. However, the injected noise needed for smoothing out the effect of binary nonlinearities often poses a challenge for optimization [28]. How noise, which is found ubiquitously in neurobiology, influences learning in the brain remains an open question.

Local Models of Computation

The locality of computations is characterized by the set variables available to the physical processing elements and depends on the computational substrate. To illustrate the concept of locality, we assume two neurons, A and B , and would like neuron A to implement a function on domain D , defined as: $D = D_{loc} \cup D_{nloc}$, where $D_{loc} = \{W_{BA}, S_A(t), U_A(t)\}$.

Here, $S^B(t-T)$ refers to the output of neuron B , T seconds ago, U_A and U_B are the respective membrane potentials, and W_{BA} is the synaptic weight from B to A , as shown in Figure S3. Variables under D_{loc} are directly available to neuron A and are thus local to it.

Conversely, variable $S^B(t-T)$ is temporally nonlocal and U_B is spatially nonlocal to neuron A . Although locality in a model of computation can make its use challenging, it enables massively parallel computations with dynamical interprocess communications.

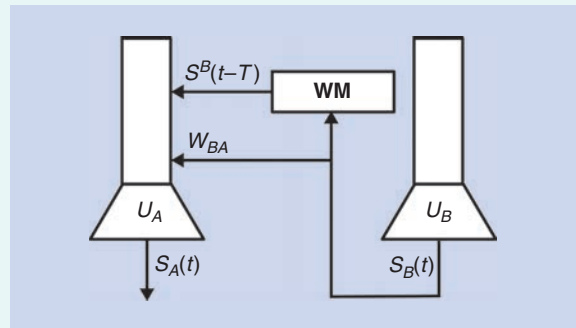


FIGURE S3. Nonlocal information can be transmitted through special structures, e.g., dedicated encoders and decoders for U_B and a form of working memory (WM) for $S^B(t-T)$.

Gradients in rate-coding networks

Another common approach to obtain gradients in SNNs is to assume a rate-based coding scheme. The main idea is that spike rate is the underlying information-carrying quantity. For many plausible neuron models, the suprathreshold firing rate depends smoothly on the neuron input. This input-output dependence is captured by the so-called f-I curve of a neuron. In such cases, the derivative of the f-I curves is suitable for gradient-based optimization.

There are several examples of this approach. For instance, Hunsberger and Eliasmith [30] as well as Neftci et al. [31] used an effectively rate-coded input scheme to demonstrate competitive performance on standard machine-learning benchmarks, such as CIFAR-10 and MNIST. Similarly, Lee et al. [32] demonstrated deep learning in SNNs by defining partial derivatives on low-pass filtered spike trains.

Rate-based approaches can offer good performance, but they may be inefficient. On the one hand, the precise estimation of firing rates requires averaging over a number of spikes. Such averaging requires either relatively high firing rates or long averaging times because several repeats are needed to average out discretization noise. This problem can be partially addressed by spatial averaging over large populations of spiking neurons. However, this may require the use of larger neuron numbers.

Finally, the distinction between rate coding and probabilistic networks can be blurry because many probabilistic network implementations use rate coding at the output level. Both types of models are differentiable, but for different reasons: Probabilistic models are based on a firing probability densities [24]. Importantly, the firing probability of a neuron is a continuous function. Although measuring probability changes requires “trial averaging” over several samples, it is the underlying continuity of the probability density that formally allows for defining differential improvements and thus, for deriving gradients. By exploiting this feature, probabilistic models have been used to learn precise output spike timing [24], [25]. In contrast, deterministic networks always emit a fixed-integer number of spikes for a given input. To nevertheless get at a notion of differential improvement, one may consider the number of spikes over a given time interval within single trials. When averaging over sufficiently large intervals, the resulting firing rates behave as a quasi-continuous function of the input current. This smooth input-output relationship is captured by the neuronal f-I curve, which can be used for optimization [30], [31]. Operating at the level of rates, however, comes at the expense of temporal precision.

Gradients in single-spike timing-coding networks

In an effort to optimize SNNs without potentially harmful noise injection and without reverting to a rate-based coding scheme, several studies have considered the outputs of neurons in SNNs to be a set of firing times. In such a temporal coding setting, individual spikes could carry significantly more information than rate-based schemes that consider only the total number of spikes in an interval.

The idea behind training temporal coding networks was pioneered in SpikeProp [33]. For this article, the analytic

expressions of firing times for hidden units were linearized, allowing for the analytical computing of approximate hidden-layer gradients. More recently a similar approach, devoid of the need for linearization, was used in [34], where the author computed the spike-timing gradients explicitly for non-LIF neurons. Intriguingly, the work showed competitive performance on conventional networks and benchmarks.

Although the spike-timing formulation does, in some cases, yield well-defined gradients, it may suffer from certain limitations. For instance, the formulation of SpikeProp [33] required each hidden unit to emit exactly one spike per trial because it is impossible to define firing time for quiescent units. Ultimately, such a nonquiescence requirement could be in conflict with power efficiency, for which it is conceivably beneficial to, e.g., have only a subset of neurons active for any given task.

Surrogate gradients

SG methods provide an alternative approach for overcoming the difficulties associated with the discontinuous nonlinearity. Moreover, they offer opportunities to reduce the potentially high algorithmic complexity associated with training SNNs. Their defining characteristic is that, instead of changing the model definition as in the smoothed approaches, an SG is introduced. In this section, we make two distinctions. We first consider SGs, which constitute a continuous relaxation of the nonsmooth spiking nonlinearity for purposes of numerical optimization (Figure 4). Such SGs do not explicitly change the optimization algorithm itself and can be used, e.g., in combination with BPTT. Further, we also consider SGs with more profound changes that explicitly affect locality of the underlying optimization algorithms themselves to improve the computational and/or memory access overhead of the learning process. One example of this approach that we will discuss involves replacing the global loss by a number of local loss functions. Finally, the use of SGs allows for the efficient end-to-end training of SNNs without needing to specify which coding scheme is to be used in the hidden layers.

Similar to standard gradient-descent learning, SG learning can deal with the spatial and temporal credit assignment problem by either BPTT or forward methods, e.g., through the use of eligibility traces (see the “Methods for Training RNNs” section for details). Alternatively, additional approximations, which may offer advantages specifically for hardware implementations, can be introduced. In the following section, we briefly review existing work that relies on SG methods before focusing on a more in-depth treatment of the underlying principles and capabilities.

In the example in Figure 4(a), we linearly interpolated between the random initial and final (postoptimization) weight matrices of the hidden-layer inputs $\mathbf{W}^{(1)}$ (network details: two input, two hidden, and two output units trained on a binary classification task). Note that the loss function [gray in Figure 4(a)] displays characteristic plateaus with a zero gradient, which is detrimental for numerical optimization.

As shown in Figure 4(b), to perform numerical optimization in this network, we constructed an SG (violet) which, in contrast

to the true gradient (gray), is nonzero. Note that we obtained the “true gradient” via the finite differences method, which, in itself, is an approximation. Importantly, the SG approximates the true gradient but retains favorable properties for optimization, i.e., continuity and finiteness. The SG can be thought of as the gradient of a virtual surrogate loss function [the violet curve in (a) obtained by numerical integration of the SG and scaled to match loss at the initial and final points]. This surrogate loss remains virtual because it is generally not computed explicitly. In practice, suitable SGs are obtained directly from the gradients of the original network through sensible approximations. This is a key difference with respect to some other approaches [22], in which the entire network is replaced explicitly by a surrogate network on which gradient descent can be performed using its true gradients.

Surrogate derivatives for the spiking nonlinearity

A set of works have used SG to specifically overcome the challenge of discontinuous spiking nonlinearity. In these works, typically, a standard algorithm such as BPTT is used with one minor modification: within the algorithm, each occurrence of the spiking nonlinearity derivative is replaced by the derivative of a continuously differentiable function. Implementing these approaches is straightforward in most autodifferentiation-enabled machine-learning toolkits.

One of the first uses of such an SG is described by Bohte in [19], where the derivative of a spiking neuron nonlinearity was approximated by the derivative of a truncated quadratic function, thus resulting in a rectifying linear unit as the surrogate derivative, as shown in Figure 3. This is similar in spirit

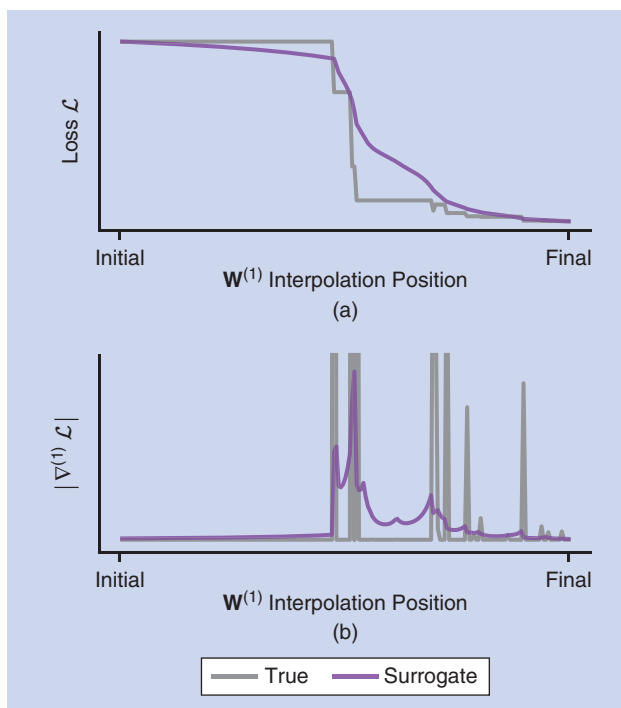


FIGURE 4. An empirical comparison of gradients and SG in an SNN. (a) The value of the loss function (gray) of an SNN classifier along an interpolation path over the hidden-layer parameters $W^{(1)}$. (b) The absolute value of the hidden-layer SG along the interpolation path.

to the solution proposed to optimize binary NNs [16]. The same idea underlies the training of large-scale convolutional networks with binary activations on classification problems using neuromorphic hardware [18]. Zenke and Ganguli [2] proposed a three-factor online learning rule using a fast sigmoid to construct an SG. Shrestha and Orchard [13] used an exponential function and reported competitive performance on a range of neuromorphic benchmark problems. Additionally, O’Connor et al. [35] described a spike-based encoding method inspired by sigma-delta modulators. They used their method to approximately encode both the activations and errors in standard feedforward artificial NNs (ANNs), and apply standard BP on these sparse-approximate encodings.

Surrogate derivatives have also been used to train spiking RCNNs, where dynamical recurrence arises due to the use of LIF neurons as well as recurrent synaptic connections. Recently, Bellec et al. [3] successfully trained RCNNs with slow temporal neuronal dynamics using a piecewise linear surrogate derivative. Encouragingly, the authors found that such networks can perform on par with conventional long short-term memory networks. Similarly, Woźniak et al. [36] reported comparable performance on a series of temporal benchmark data sets.

In summary, a plethora of studies have constructed SG using different nonlinearities and trained a diversity of SNN architectures. These nonlinearities, however, have a common underlying theme: All functions are nonlinear and monotonically increase toward the firing threshold, as shown in Figure 3. Although a more systematic comparison of different surrogate nonlinearities is still pending, overall, the diversity found in current literature suggests that the success of the method is not crucially dependent on the details of the surrogate used to approximate the derivative.

Surrogate gradients that affect the locality of update rules

The majority of studies discussed in the previous section introduced a surrogate nonlinearity to prevent gradients from vanishing (or exploding), but, by relying on methods such as BPTT, they did not explicitly affect the structural properties of the learning rules. There are, however, training approaches for SNNs that introduce more far-reaching modifications, which may completely alter the way error or target signals are propagated (or generated) within the network. Such approaches are typically used in conjunction with the aforementioned surrogate derivatives. There are two main motivations for such modifications, which are typically linked to physical constraints that make it impossible to implement the “correct” gradient-descent algorithm. For instance, in neurobiology, biophysical constraints make it impossible to implement BPTT without further approximations. Studies interested in how the brain could solve the credit assignment problem focus on how simplified “local” algorithms could achieve similar performance while adhering to the constraints of the underlying biological wetware (see “Local Models of Computation”). Similarly, neuromorphic hardware may pose certain constraints with regard to memory or communications, which impede the use of BPTT and call for simpler and often more local methods for training on such devices.

As training SNNs using SGs advances to deeper architectures, it is foreseeable that additional problems, similar to the ones encountered in ANNs, will arise. For example, several approaches currently rely on SGs derived from sigmoidal activation functions, as shown in Figure 3. The use of sigmoidal activation functions, however, is associated with vanishing gradient problems. Another set of challenges, which may need addressing in the future, could be linked to the bias that SGs introduce into the learning dynamics.

In the following section, we review a selection of promising SG approaches, which introduce far larger deviations from the “true gradients” and still allow for learning at a greatly reduced complexity and computational cost.

Applications

In this section, we present a selection of illustrative applications of smooth or SGs to SNNs, which exploit both the internal continuous-time dynamics of the neurons and their event-driven nature. The latter allows a network to remain quiescent until incoming spikes trigger activity.

Feedback alignment and random error BP

One family of algorithms that relaxes some of the requirements of BP is feedback alignment or, more generally, random BP algorithms [Figure 5(b) and (c)] [37]–[39]. These are approximations to the gradient BP rule that sidestep the nonlocality problem by replacing weights in the BP rule with random ones, as shown in Figure 5(b): $\delta_i^{(l)} = \sigma'(a_i^{(l)}) \sum_k \delta_k^{(l+1)} G_{ki}^{(l)}$, where $\mathbf{G}^{(l)}$ is a fixed, random matrix with the same dimensions as \mathbf{W} . The replacement of $\mathbf{W}^{T,(l)}$ with a random matrix $\mathbf{G}^{(l)}$ breaks the dependency of the backward phase on $\mathbf{W}^{(l)}$, enabling the rule to be more local. One common variation is to replace the entire backward propagation by a random propagation of the errors to each layer, as depicted in Figure 5(c) [38]: $\delta_i^{(l)} = \sigma'(a_i^{(l)}) \sum_k \delta_k^{(L)} H_{ki}^{(l)}$, where $\mathbf{H}^{(l)}$ is a fixed, random matrix with appropriate dimensions.

Random BP approaches lead to remarkably little loss in classification performance on some benchmark tasks. Although a general theoretical understanding of random BP is still a subject of intense research, simulation studies have shown that, during learning, the network adjusts its feedforward weights such that they partially align with the (random) feedback weights, thus permitting them to convey useful error information [37]. Building on these findings, an asynchronous spike-driven adaptation of random BP using local synaptic plasticity rules with the dynamics of spiking neurons was demonstrated in [31]. To obtain SGs, the authors approximated the derivative of the neural activation function using a symmetric function that is zero everywhere except in the vicinity of zero, where it is constant. Networks using this learning rule performed remarkably well, and were shown to operate continuously and asynchronously without the alternation between forward and backward passes, which is necessary in BP. One important limitation with random BP applied to SNNs was that the temporal dynamics of the neurons and synapses was not taken into account in the gradients. SuperSpike solves this problem.

Supervised learning with local three-factor learning rules

SuperSpike is a biologically plausible three-factor learning rule. In contrast to many existing three-factor rules that fall into the category of “smoothed approaches” [24]–[29], SuperSpike is an SG approach that combines several approximations to render it more biologically plausible [2]. Although the underlying motivation of the study is geared toward a deeper understanding of learning in biological NNs, the learning rule may prove interesting for hardware implementations because it is an online rule that does not require backpropagating error information through time. Specifically, the rule uses synaptic eligibility traces to solve the temporal credit assignment problem.

The SuperSpike learning rule is a forward-in-time optimization procedure that was derived for temporal supervised learning tasks in which a given output neuron learns to spike at predefined times. To that end, it minimizes the van Rossum distance with kernel λ between a set of output spike trains $S_i(t)$ and their corresponding target spike trains $S_i^*(t)$

$$\mathcal{L} = \frac{1}{2} \int_{-\infty}^t \sum_i (\lambda * (S_i[s] - S_i^*[s]))^2 ds \approx \frac{1}{2} \sum_{n,k} \underbrace{(\lambda * (S_i[n] - S_i^*[n]))^2}_{\equiv e_i^2[n]}, \quad (7)$$

where the last approximation corresponds to transitioning to discrete time. To avoid nonlocality, SuperSpike relies on a form of random BP to propagate error signals directly from the output layer to the hidden units. In deep networks, we expect this coarse approximation to cause problems for learning. In such cases, it may be important to compensate for layer-specific delays or to use entirely different approaches for credit assignment (compare the “Learning Using Local Errors” section). Because hidden layers use the same learning rule as the output layer, in the following section, we focus on a network without hidden layers to illustrate the online character of the rule.

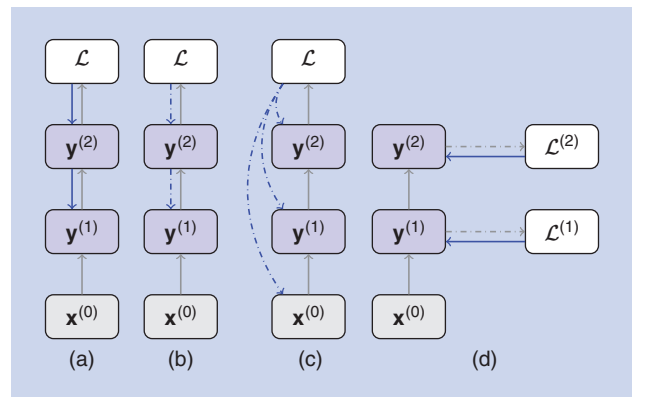


FIGURE 5. The strategies for relaxing gradient BP requirements. The dashed lines indicate fixed, random connections. (a) BP propagates errors through each layer using the transpose of the forward weights by alternating forward and backward passes. (b) FA [37] replaces the transposed matrix with a random one. (c) DFA [38] directly propagates the errors from the top layer to the hidden layers. (d) Local errors [29] uses a fixed, random, auxiliary cost function at each layer.

To perform online gradient descent on \mathcal{L} , we compute the gradients of the squared output error signals $e_i^2[n]$ at each time step n . Here, we first encounter the derivative $(\partial/\partial W_{ij})\lambda * S_i[n]$. Because the (discrete) convolution is a linear operator, this expression simplifies to $\lambda * (\partial S_i[n]/\partial W_{ij})$. To compute derivatives of the neuron's output spike train of the form $\partial S_i[n]/\partial W_{ij}$, we differentiate the network dynamics, i.e., (4) and (5), and obtain

$$\frac{\partial S_i[n+1]}{\partial W_{ij}} = \Theta'(U_i[n+1] - \vartheta) \left[\frac{\partial U_i[n+1]}{\partial W_{ij}} \right], \quad (8)$$

$$\frac{\partial U_i[n+1]}{\partial W_{ij}} = \beta \frac{\partial U_i[n]}{\partial W_{ij}} + \frac{\partial I_i[n]}{\partial W_{ij}} - \frac{\partial S_i[n]}{\partial W_{ij}}, \quad (9)$$

$$\frac{\partial I_i[n+1]}{\partial W_{ij}} = \alpha \frac{\partial I_i[n]}{\partial W_{ij}} + S_j[n]. \quad (10)$$

Equations (8)–(11) define a dynamical system which, given the starting conditions $S_i[0] = U_i[0] = I_i[0] = 0$, can be simulated online and forward in time to produce all relevant derivatives. Importantly, the convolution with λ is implemented similarly to (9) and (10) as a double integrator (see [2]). These equations are conceptually similar to those derived under RTRL, i.e., (6). Crucially, to arrive at useful SGs, SuperSpike makes two approximations. First, Θ' is replaced by a smooth surrogate derivative $\sigma'(U[n] - \vartheta)$ (compare Figure 3). Second, the reset term with the negative sign in (9) is dropped, which empirically leads to better results. With these definitions in hand, the final weight updates are given by

$$\Delta W_{ij}[n] \propto e_i[n] \lambda * \left[\sigma'(U_i[n] - \vartheta) \frac{\partial U_i[n]}{\partial W_{ij}} \right], \quad (11)$$

where $e_i[n] \equiv \lambda * (S_i - S_i^*)$. These weight updates depend only on local quantities and error signals (see “Local Models of Computation”).

So far, in this section, we have considered a simple two-layer network (compare Figure 2) without recurrent connections. If we were to apply the same strategy to compute updates in an RCNN or a network with an additional hidden layer, the equations would become more complicated and nonlocal. SuperSpike, when applied to multilayer networks, sidesteps this issue by propagating error signals from the output layer directly to the hidden units, as in random BP (compare the “Feedback Alignment and Random Error BP” section) Figure 5(c), [37]–[39]. For networks with additional hidden layers, the output errors are simply broadcast through either random or structured weights A :

$$\Delta W_{ij}^{(l)}[n] \propto \left[\sum_k A_{ik}^{(l)} e_k[n] \right] \lambda * \left[\sigma'(U_i^{(l)}[n] - \vartheta) \frac{\partial U_i^{(l)}[n]}{\partial W_{ij}^{(l)}} \right]. \quad (12)$$

Thus, SuperSpike achieves temporal credit assignment by propagating all relevant quantities forward in time through eligibility traces defined by the neuronal dynamics [(9) and (10)], while it relies on random BP to perform spatial credit assignment.

Although the work by Zenke and Ganguli [2] was centered around feedforward networks, Bellec et al. [15] show that similar biologically plausible three factors rule can also be used to train RCNNs efficiently.

Learning using local errors

In practice, the performance of SuperSpike does not scale favorably for large multilayer networks. The scalability of SuperSpike can be improved by introducing local errors, as described in this section.

Multilayer NNs are hierarchical feature extractors. Through successive linear projections and pointwise nonlinearities, neurons become tuned (i.e., respond most strongly) to particular spatiotemporal features in the input. Although the best features are those that take into account the subsequent processing stages, and which, are learned to minimize the final error (as the features learned using BP do), high-quality features can also be obtained by more local methods. The nonlocal component of the weight update, i.e., (S1), is the error term $\delta_i^{(l)}[n]$. Rather than obtaining this error term through BP, it can be generated using information local to the layer. One way of achieving this is to define a layerwise loss $\mathcal{L}^{(l)}(\mathbf{y}^{(l)}[n])$ and use this local loss to obtain the errors. In such a local learning setting, the local errors $\delta^{(l)}$ become

$$\delta_i^{(l)}[n] = \sigma'(a_i^{(l)}[n]) \frac{d}{dy_i^{(l)}[n]} \mathcal{L}^{(l)}(\mathbf{y}^{(l)}[n]),$$

where

$$\mathcal{L}^{(l)}(\mathbf{y}^{(l)}[n]) \equiv \mathcal{L}(\mathbf{G}^{(l)} \mathbf{y}^{(l)}[n], \hat{\mathbf{y}}^{(l)}[n]), \quad (13)$$

with $\hat{\mathbf{y}}^{(l)}[n]$ a pseudotarget for layer l , and $\mathbf{G}^{(l)}$ a fixed random matrix that projects the activity vector at layer l to a vector having the same dimension as the pseudotarget. In essence, this formulation assumes that an auxiliary random layer is attached to layer l , with the goal of modifying $\mathbf{W}^{(l)}$ so as to minimize the discrepancy between the auxiliary random layer's output and the pseudotarget. The simplest choice for the pseudotarget is to use the top-layer target. This forces each layer to learn a set of features that can match the top-layer target after undergoing a fixed random linear projection. Each layer builds on the features learned by the layer below it, and we empirically observe that higher layers are able to learn higher-quality features that allow their random and fixed auxiliary layers to better match the target [40].

A related approach was explored with SNNs [41], where separate networks provided high-dimensional temporal signals that improve learning. Local errors were recently used in SNNs in combination with the SuperSpike (compare the “Supervised Learning With Local Three-Factor Learning Rules” section) forward method to overcome the temporal credit assignment problem [4]. As in SuperSpike, the SNN model is simplified by using a feedforward structure and by omitting the refractory dynamics in the optimization; however, the cost function was defined to operate locally on the instantaneous rates of each layer. This simplification results in a forward method whose space complexity scales as $O(N)$ [rather than $O(N^3)$].

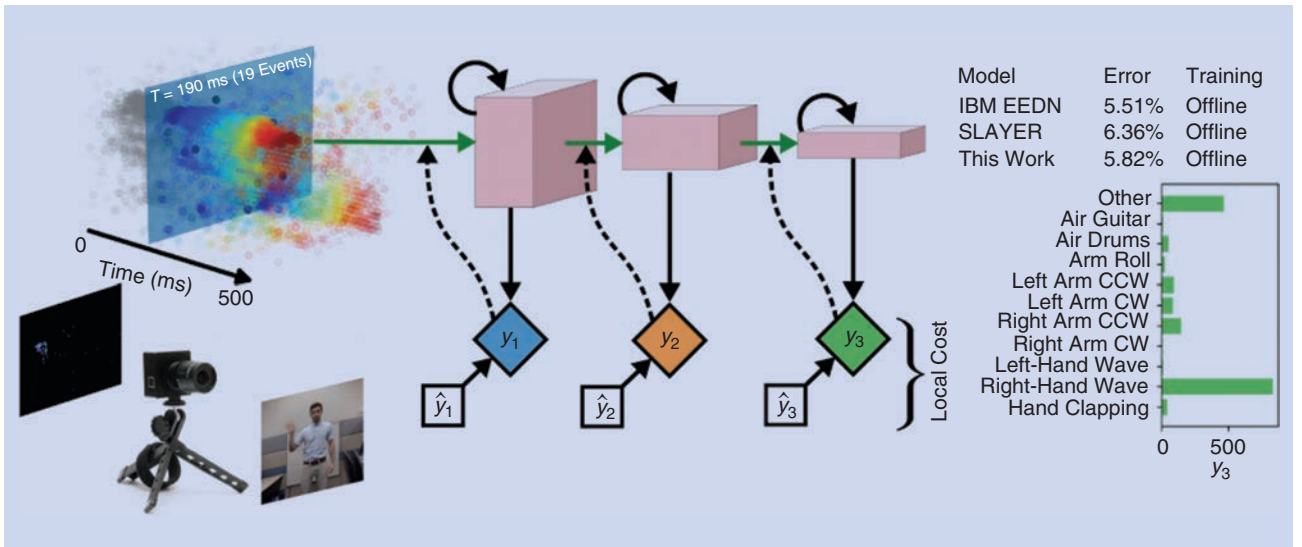


FIGURE 6. DECOLLE with spikes [4] applied to the event-based DVS Gestures data set. The feedforward weights (green) of a three-layer convolutional SNN are trained with SG; local errors are generated using fixed random projections onto a local classifier. Learning in DECOLLE scales linearly with the number of neurons, thanks to local rate-based cost functions formed by spike-based basis functions. The circular arrows indicate recurrence due to the statefulness of the LIF dynamics (no recurrent synaptic connections were used here) and are not trained. This SNN outperforms BPTT methods [13] and requires fewer training iterations [4] compared to other approaches. SLAYER: Spike Layer Error Reassignment [20]; EEDN: energy-efficient deep neuromorphic networks.

for the forward method, $O(N^2)$ for SuperSpike, or $O(NT)$ for the backward method], while still making use of spiking neural dynamics. Thus, the method constitutes a highly efficient synaptic plasticity rule for multilayer SNNs. Furthermore, the simplifications enable the use of existing automatic differentiation methods in machine-learning frameworks to systematically derive synaptic plasticity rules from task-relevant cost functions and neural dynamics (see [4] and included tutorials), thereby making deep continuous local learning (DECOLLE) easy to implement. This approach was benchmarked on the dynamic vision sensor (DVS) Gestures data set (Figure 6), and performs on par with standard BP or BPTT rules.

Learning using gradients of spike times

Difficulties in training SNNs stem from the discrete nature of the quantities of interest, such as the number of spikes in a particular interval. The derivatives of these discrete quantities are zero almost everywhere, which necessitates the use of SG methods. Alternatively, we can choose to use spike-based quantities that have well-defined, smooth derivatives. One such quantity is spike times. This capitalizes on the continuous-time nature of SNNs and results in highly sparse network activity, as the emission time of even a single spike can encode significant information. Just as importantly, spike times are continuous quantities that can be made to depend smoothly on the neuron's input. Working with spike times is thus a complementary approach to SG and achieves the same goal: obtaining a smooth chain of derivatives between the network's outputs and inputs. For this example, we use nonleaky neurons described by

$$\frac{dU_i}{dt} = I_i \quad \text{with} \quad I_i = \sum_j W_{ij} \sum_r \Theta(t - t'_j) \exp(-(t - t'_j)), \quad (14)$$

where t'_j is the time of the r th spike from neuron j and Θ is the Heaviside step function.

Consider the simple exclusive or problem in the temporal domain: a network receives two spikes, one from each of two different sources. Each spike can either be “early” or “late.” The network must learn to distinguish between the case in which the spikes are either both early or both late, and the case where one spike is early and the other is late, as shown in Figure 7(a). When designing an SNN, there is significant freedom in how the network input and output are encoded. In this case, we use a first-to-spike code in which we have two output neurons, and the binary classification result is represented by the output neuron that spikes first. Figure 7(b) shows the network's response after training (see [34] for details on the training process). For the first input class (early/late or late/early), one output neuron spikes first, and for the other class (early/early or late/late), the other output neuron spikes first.

Conclusions

We have outlined how discrete-time SNNs can be studied within the framework of RNNs and discussed successful approaches for training them. We have specifically focused on SG approaches for two reasons: SG approaches are able to train SNNs to perform at unprecedented performance levels on a range of real-world problems. This transition marks the beginning of an exciting time in which SNNs will garner increasing interest for applications that were previously dominated by nonspiking RNNs; SGs provide a framework that ties together ideas from machine learning, computational neurosciences, and neuromorphic computing. We emphasize that, although SGs are well defined in the discrete-time

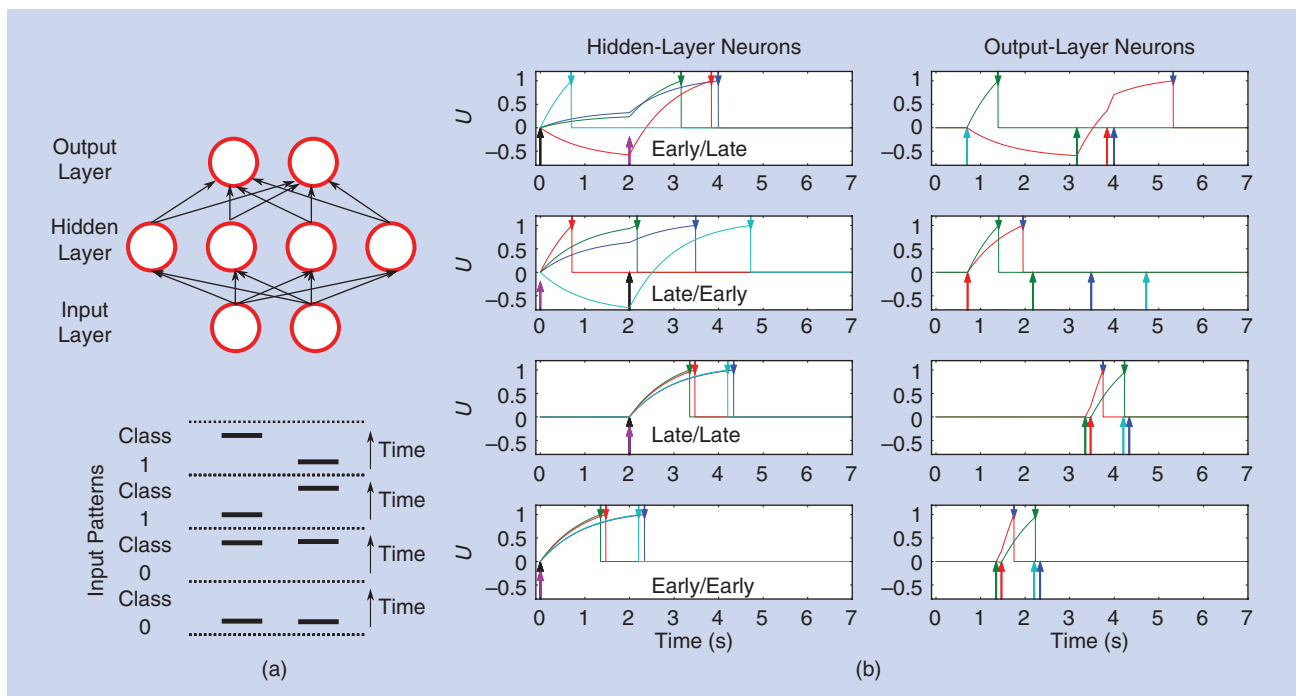


FIGURE 7. The temporal exclusive or problem. (a) An SNN with one hidden layer. Each input neuron emits one spike, which can either be late or early, resulting in four possible input patterns that should be classified into two classes. (b) For the four input spike patterns (one per row), the right plots show the membrane potentials of the two output neurons, while the left plots show the membrane potentials of the four hidden neurons. Arrows at the top of the plot indicate output spikes from the layer, while arrows at the bottom indicate input spikes. The output spikes of the hidden layer are the input spikes of the output layer. The classification result is encoded in the identity of the output neuron that spikes first.

framework studied in this article, the theoretical foundations of SGs for SNNs remain an open problem, including the generalization of spike-based BPTT to continuous-time dynamics and the optimal choice of smooth activation functions. From the viewpoint of computational neuroscience, the approaches presented in this article are appealing because several of them are related to “three-factor” plasticity rules, which are an important class of rules believed to underlie synaptic plasticity in the brain. Finally, for the neuromorphic community, SG methods provide a way to learn under various constraints on communication and storage, which makes SG methods highly relevant for learning on customized, low-power neuromorphic devices.

The spectacular successes of modern ANNs were enabled by algorithmic and hardware advances that made it possible to efficiently train large ANNs on vast amounts of data. With temporal coding, SNNs are universal function approximators that are potentially far more powerful than ANNs with sigmoidal nonlinearities. Unlike large-scale ANNs, which had to wait for several decades until the necessary computational resources were available for training them, we currently have the necessary resources, whether in the form of mainstream compute devices such as CPUs or GPUs, or custom neuromorphic devices, to train and deploy large SNNs. The fact that SNNs are less widely used than ANNs is thus primarily due to the algorithmic issue of trainability. In this article, we provided an overview of various exciting developments that are gradually addressing the issues

encountered when training SNNs. Fully addressing these issues would have immediate and wide-ranging implications, both technologically and in relation to learning in biological brains.

Acknowledgments

This work was supported by the Intel Corporation (to Emre Neftci), the National Science Foundation under grant 1640081 (to Emre Neftci), the Swiss National Science Foundation Early Postdoc Mobility Grant P2ZHP2_164960 (to Hesham Mostafa), and the Wellcome Trust [110124/Z/15/Z] (to Friedemann Zenke).

Authors

Emre O. Neftci (eneftci@uci.edu) received his M.Sc. degree in physics from École Polytechnique Fédérale de Lausanne, Switzerland, and his Ph.D. degree in neuroinformatics from the Institute of Neuroinformatics at the University of Zürich and ETH Zürich, in 2010. Currently, he is an assistant professor in the Department of Cognitive Sciences and Computer Science at the University of California, Irvine. His current research explores the bridges between neuroscience and machine learning, with a focus on the theoretical and computational modeling of learning algorithms that are best suited to neuromorphic hardware and non-von Neumann computing architectures. He is a Member of the IEEE.

Hesham Mostafa (hesham.mostafa@intel.com) received his M.Sc. degree in electrical engineering from the Technical

University of Munich, Germany, in 2010 and his Ph.D. degree in neuroinformatics from the Institute of Neuroinformatics at the University of Zürich and ETH Zürich in 2016. Currently, he is a research scientist in the office of the CTO with Intel's Artificial Intelligence Products Group. His research interests include combining ideas from machine learning and computational neuroscience for developing biologically inspired and hardware-efficient learning and optimization algorithms, and physically implementing these algorithms using CMOS and novel device technologies.

Friedemann Zenke (friedemann.zenke@fmi.ch) received his diploma in physics from the University of Bonn, Germany, and the Australian National University, Canberra, in 2009 and received his Ph.D. degree from the École Polytechnique Fédérale de Lausanne, Switzerland, on the interaction of synaptic and homeostatic plasticity in spiking neural network models, in 2014. Currently, he is a junior group leader at the Friedrich Miescher Institute for Biomedical Research, Basel, Switzerland. His research interests include studying learning in biologically inspired network models with a focus on deep credit assignment and unsupervised learning.

References

- [1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.
- [2] F. Zenke and S. Ganguli, "SuperSpike: Supervised learning in multilayer spiking neural networks," *Neural Comput.*, vol. 30, no. 6, pp. 1514–1541, 2018.
- [3] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Red Hook, New York: Curran Associates, 2018, pp. 795–805.
- [4] J. Kaiser, H. Mostafa, and E. Neftci, "Synaptic plasticity for deep continuous local learning," 2018. [Online]. Available: arxiv:1811.10766
- [5] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep learning in spiking neural networks," *Neural Netw.*, vol. 111, pp. 47–63, Mar. 2019.
- [6] R. Güttig, "To spike, or when to spike?" *Curr. Opin. Neurobiol.*, vol. 25, pp. 134–139, Apr. 2014.
- [7] R.-M. Memmesheimer, R. Rubin, B. Ölveczky, and H. Sompolinsky, "Learning precisely timed spikes," *Neuron*, vol. 82, no. 4, pp. 925–938, 2014.
- [8] N. Anwani and B. Rajendran, "NormAD-normalized approximate descent based supervised learning rule for spiking neurons," in *Proc. IEEE Int. Joint Conf. Neural Networks (IJCNN)*, 2015, pp. 1–8. doi: 10.1109/IJCNN.2015.7280618.
- [9] A. Gilra and W. Gerstner, "Predicting non-linear dynamics by stable local learning in a recurrent spiking neural network," *eLife*, vol. 6, Nov. 2017. doi: 10.7554/eLife.28295, [Online]. Available: <https://elifesciences.org/articles/28295>
- [10] W. Nicola and C. Clopath, "Supervised learning in spiking neural networks with FORCE training," *Nat. Commun.*, vol. 8, Dec. 2017. doi: 10.1038/s41467-017-01827-3, [Online]. Available: <https://www.nature.com/articles/s41467-017-01827-3>
- [11] K. Boahen, "A neuromorph's prospectus," *Comput. Sci. Eng.*, vol. 19, no. 2, pp. 14–28, 2017.
- [12] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge, U.K.: Cambridge Univ. Press, 2014.
- [13] S. B. Shrestha and G. Orchard, "SLAYER: Spike layer error reassignment in time," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Red Hook, New York: Curran Associates, 2018, pp. 1419–1428.
- [14] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, 1989.
- [15] G. Bellec, F. Scherr, E. Hajek, D. Salaj, R. Legenstein, and W. Maass, "Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets," 2019. [Online]. Available: arxiv:1901.09049
- [16] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," 2016. [Online]. Available: arxiv:1602.02830
- [17] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013. [Online]. Available: arxiv:1308.3432
- [18] S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry et al., "Convolutional networks for fast, energy-efficient neuromorphic computing," *Proc. Nat. Acad. Sci.*, vol. 113, no. 41, pp. 11,441–11,446, 2016.
- [19] S. M. Bohte, "Error-backpropagation in networks of fractionally predictive spiking neurons," in *Proc. Int. Conf. Artificial Neural Networks (ICANN)*, 2011, 60–68.
- [20] S. B. Shrestha and G. Orchard, "SLAYER: Spike layer error reassignment in time," in *Advances in Neural Information Processing Systems*, vol. 31, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Red Hook, New York: Curran Associates, 2018, pp. 1412–1421. [Online]. Available: <http://papers.nips.cc/paper/7415-slayer-spike-layer-error-reassignment-in-time.pdf>
- [21] L. F. Abbott, B. DePasquale, and R.-M. Memmesheimer, "Building functional networks of spiking model neurons," *Nat. Neurosci.*, vol. 19, no. 3, pp. 350–355, 2016.
- [22] D. Huh and T. J. Sejnowski, "Gradient descent for spiking neural networks," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Red Hook, New York: Curran Associates, 2018, pp. 1440–1450.
- [23] D. Ackley, G. Hinton, and T. Sejnowski, "A learning algorithm for Boltzmann machines," *Cogn. Sci.: A Multidisciplinary J.*, vol. 9, no. 1, pp. 147–169, 1985.
- [24] J.-P. Pfister, T. Toyozumi, D. Barber, and W. Gerstner, "Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning," *Neural Comput.*, vol. 18, no. 6, pp. 1318–1348, 2006.
- [25] B. Gardner, I. Sporea, and A. Grüning, "Learning spatiotemporally encoded pattern transformations in structured spiking neural networks," *Neural Comput.*, vol. 27, no. 12, pp. 2548–2586, 2015.
- [26] J. Guerguiev, T. P. Lillicrap, and B. A. Richards, "Towards deep learning with segregated dendrites," *eLife*, vol. 6, Dec. 2017. doi: 10.7554/eLife.22901, [Online]. Available: <https://elifesciences.org/articles/22901>
- [27] J. Brea, W. Senn, and J.-P. Pfister, "Matching recall and storage in sequence learning with spiking neural networks," *J. Neurosci.*, vol. 33, no. 23, pp. 9565–9575, 2013.
- [28] D. J. Rezende and W. Gerstner, "Stochastic variational learning in recurrent spiking networks," *Front. Comput. Neurosci.*, vol. 8, p. 38, April 2014. doi: 10.3389/fncom.2014.00038.
- [29] H. Mostafa and G. Cauwenberghs, "A learning framework for winner-take-all networks with stochastic synapses," *Neural Comput.*, vol. 30, no. 6, pp. 1542–1572, 2018.
- [30] E. Hunsberger and C. Eliasmith, "Spiking deep networks with LIF neurons," 2015. [Online]. Available: arxiv:1510.08829
- [31] E. O. Neftci, C. Augustine, S. Paul, and G. Detorakis, "Event-driven random back-propagation: Enabling neuromorphic deep learning machines," *Front. Neurosci.*, vol. 11, p. 324, June 2017.
- [32] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Front. Neurosci.*, vol. 10, Nov. 2016. doi: 10.3389/fnins.2016.00508, [Online]. Available: <https://www.frontiersin.org/articles/10.3389/fnins.2016.00508/full>
- [33] S. M. Bohte, J. N. Kok, and H. La Poutre, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, no. 1–4, pp. 17–37, 2002.
- [34] H. Mostafa, "Supervised learning based on temporal coding in spiking neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 7, pp. 3227–3235, 2018.
- [35] P. O'Connor, E. Gavves, and M. Welling, "Temporally efficient deep learning with spikes," 2017. [Online]. Available: arxiv:1706.04159
- [36] S. Woźniak, A. Pantazi, and E. Eleftheriou, "Deep networks incorporating spiking neural dynamics," 2018. [Online]. Available: arxiv:1812.07040
- [37] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nat. Commun.*, vol. 7, no. 1, 2016. doi: 10.1038/ncomms13276, [Online]. Available: <https://www.nature.com/articles/ncomms13276>
- [38] A. Nøkland, "Direct feedback alignment provides learning in deep neural networks," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Red Hook, New York: Curran Associates, 2016, pp. 1037–1045.
- [39] P. Baldi and P. Sadowski, "A theory of local learning, the learning channel, and the optimality of backpropagation," *Neural Netw.*, vol. 83, pp. 51–74, Nov. 2016.
- [40] H. Mostafa, V. Ramesh, and G. Cauwenberghs, "Deep supervised learning using local errors," *Front. Neurosci.*, vol. 12, p. 608, Aug. 2018.
- [41] W. Nicola and C. Clopath, "Supervised learning in spiking neural networks with force training," *Nat. Commun.*, vol. 8, no. 1, 2017. doi: 10.1038/s41467-017-01827-3, [Online]. Available: <https://www.nature.com/articles/s41467-017-01827-3>

An Introduction to Probabilistic Spiking Neural Networks

Probabilistic models, learning rules, and applications



©ISTOCKPHOTO.COM/JUST_SUPER

Spiking neural networks (SNNs) are distributed trainable systems whose computing elements, or neurons, are characterized by internal analog dynamics and by digital and sparse synaptic communications. The sparsity of the synaptic spiking inputs and the corresponding event-driven nature of neural processing can be leveraged by energy-efficient hardware implementations, which can offer significant energy reductions as compared to conventional artificial neural networks (ANNs). The design of training algorithms for SNNs, however, lags behind hardware implementations: most existing training algorithms for SNNs have been designed either for biological plausibility or through conversion from pretrained ANNs via rate encoding.

This article provides an introduction to SNNs by focusing on a probabilistic signal processing methodology that enables the direct derivation of learning rules that leverage the unique time-encoding capabilities of SNNs. We adopt discrete-time probabilistic models for networked spiking neurons and derive supervised and unsupervised learning rules from first principles via variational inference. Examples and open research problems are also provided.

Introduction

ANNs have become the de facto standard tool to carry out supervised, unsupervised, and reinforcement learning tasks. Their recent successes range from image classifiers that outperform human experts in medical diagnosis to machines that defeat professional players at complex games, such as Go. These breakthroughs have built upon various algorithmic advances but have also heavily relied on the unprecedented availability of computing power and memory in data centers and cloud computing platforms. The resulting considerable energy requirements run counter to the constraints imposed by implementations on low-power mobile or embedded devices for such applications as personal health monitoring or neural prosthetics [1].

ANNs versus SNNs

Various new hardware solutions have recently emerged that attempt to improve the energy efficiency of ANNs as inference

machines by trading complexity for accuracy in the implementation of matrix operations. A different line of research, which is the subject of this article, seeks an alternative framework that enables efficient online inference and learning by taking inspiration from the working of the human brain.

The human brain is capable of performing general and complex tasks via continuous adaptation at a minute fraction of the power required by state-of-the-art supercomputers and ANN-based models [2]. Neurons in the human brain are qualitatively different from those in an ANN: they are dynamic devices featuring recurrent behavior, rather than static nonlinearities, and they process and communicate using sparse spiking signals over time, rather than real numbers. Inspired by this observation, as illustrated in Figure 1, SNNs have been introduced in the theoretical neuroscience literature as networks of dynamic spiking neurons [3]. SNNs have the unique capability to process information encoded in the timing of events, or spikes. Spikes are also used for synaptic communications, with synapses delaying and filtering signals before they reach the postsynaptic neuron. Because of the presence of synaptic delays, neurons in an SNN can be naturally connected via arbitrary recurrent topologies, unlike standard multilayer ANNs or chain-like recurrent neural networks.

Proof-of-concept and commercial hardware implementations of SNNs have demonstrated orders-of-magnitude improvements in terms of energy efficiency over ANNs [4]. Given the extremely low idle energy consumption, the energy spent by SNNs for learning and inference is essentially proportional to the number of spikes processed and communicated by the neurons, with the energy per spike being as low as a few picojoules [5].

Deterministic versus probabilistic SNN models

The most common SNN model consists of a network of neurons with deterministic dynamics whereby a spike is emitted as soon as an internal state variable, known as the *membrane potential*, crosses a given threshold value. A typical example is the leaky integrate-and-fire model, in which the membrane potential increases with each spike recorded in the incoming synapses while decreasing in the absence of inputs. When information is encoded in the rate of spiking of the neurons, an SNN can approximate the behavior of a conventional ANN with the same topology. This has motivated a popular line of work that aims at converting a pretrained ANN into a potentially more efficient SNN implementation (see [6] and the “Models” section for further details).

To make full use of the temporal processing capabilities of SNNs, learning problems should be formulated as the minimization of a loss function that directly accounts for the timing of the spikes emitted by the neurons. As for ANNs, this minimization can, in principle, be done using stochastic gradient descent (SGD). Unlike ANNs, however, this conventional approach is made challenging by the nondifferentiability of the output of the SNN with respect to the synaptic weights due to the threshold crossing-triggered behavior of spiking neurons. The potentially complex recurrent topology of SNNs also makes it difficult to implement the standard backpropagation

procedure used in multilayer ANNs to compute gradients. To obviate this problem, a number of existing learning rules approximate the derivative by smoothing out the membrane potential as a function of the weights [7]–[9].

In contrast to deterministic models for SNNs, a probabilistic model defines the outputs of all spiking neurons as jointly distributed binary random processes. The joint distribution is differentiable in the synaptic weights, and, as a result, so are principled learning criteria from statistics and information theory, such as likelihood function and mutual information. The maximization of such criteria can apply to arbitrary topologies and does not require the implementation of backpropagation mechanisms. Hence, a stochastic viewpoint has significant analytic advantages, which translate into the derivation of flexible learning rules from first principles. These rules recover as special cases many known algorithms proposed for SNNs in the theoretical neuroscience literature as biologically plausible algorithms [10].

Scope and overview

This article aims to provide a review on the topic of probabilistic SNNs with a specific focus on the most commonly used generalized linear models (GLMs). We cover models, learning rules, and applications, highlighting principles and tools. The main goal is to make key ideas in this emerging field accessible to researchers in signal processing, who may otherwise find it difficult to navigate the theoretical neuroscience literature on the subject, given its focus on biological plausibility rather than

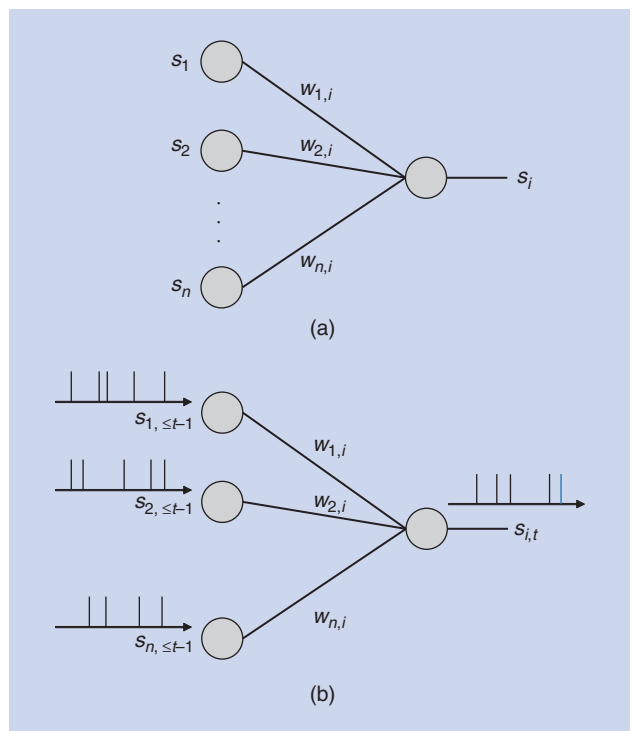


FIGURE 1. Illustrations of neural network models: (a) an ANN, where each neuron i processes real numbers s_1, \dots, s_n to output and communicates a real number s_i as a static nonlinearity, and (b) an SNN, where dynamic spiking neurons process and communicate sparse spiking signals over time t in a causal manner to output and communicate a binary spiking signal $s_{i,t}$.

theoretical and algorithmic principles [10]. At the end of the article, we also review alternative probabilistic formulations of SNNs, extensions, and open problems.

Learning tasks

An SNN is a network of spiking neurons. As seen in Figure 2, the input and output interfaces of an SNN typically transfer spiking signals. Input spiking signals can either be recorded directly from neuromorphic sensors, such as silicon cochleas and retinas [Figure 2(a)], or be converted from a natural signal to a set of spiking signals [Figure 2(b)]. Conversion can be done by following different rules, including rate encoding, whereby amplitudes are converted into the (instantaneous) spiking rate of a neuron; time encoding, in which amplitudes are translated into spike timings; and population coding, whereby amplitudes are encoded into the (instantaneous) firing rates [11] or relative firing times of a subset of neurons (see [10] for a review). In a similar manner, output spiking signals can either be fed directly to a neuromorphic actuator, such as neuromorphic controllers or prosthetic systems [Figure 2(a)], or be converted from spiking signals to natural signals [Figure 2(b)]. This can be done by following rate, time, or population decoding principles.

The SNN generally acts as a dynamic mapping between inputs and outputs that is defined by the model parameters, including, most notably, the interneuron synaptic weights. This mapping can be designed or trained to carry out inference or control tasks. When training is enabled, the model parameters are automatically adapted based on data fed to the network, with the goal of maximizing a given performance criterion. Training can be carried out in a supervised, unsupervised, or reinforcement learning manner, depending on the availability of data and feedback signals, as further discussed subsequently. For both inference/control and training, data can be presented to the SNN in a batch mode (also known as a *frame-based mode*) or in an *online mode* (see the “Training SNNs” section).

With supervised learning, the training data specify both the input and desired output. Input and output pairs are either in the form of a number of separate examples, in the case of batch

learning, or presented over time in a streaming fashion for online learning. As an example, the training set may include a number of spike-encoded images and corresponding correct labels, or a single time sequence to be used to extrapolate predictions (see also the “Batch Learning Examples” and “Online Learning Examples” sections). Under unsupervised learning, the training data specify only the desired input or output to the SNN, which can again be presented in a batch or online fashion. Examples of applications include representation learning, which aims to translate the input into a more compact, interpretable, or useful representation, and generative modeling, which seeks to generate outputs with statistics akin to the training data (see, e.g., [12]). Finally, with reinforcement learning, the SNN is used to control an agent on the basis of input observations from the environment to accomplish a given goal. To this end, the SNN is provided with feedback on the selected outputs that guides the SNN in updating its parameters in a batch or online manner [13].

Models

Here, we describe the standard discrete-time GLM for SNNs, also known as the *spike response model with escape noise* (see, e.g., [14] and [15]). Discrete-time models reflect the operation of a number of neuromorphic chips, including Intel’s Loihi [4], while continuous-time models are more commonly encountered in the computer neuroscience literature [10].

Graphical representation

As illustrated in Figure 3, an SNN consists of a network of N spiking neurons. At any time $t = 0, 1, 2, \dots$, each neuron i outputs a binary signal $s_{i,t} \in \{0, 1\}$, with value $s_{i,t} = 1$ corresponding to a spike emitted at time t . We collect in vector $\mathbf{s}_t = (s_{i,t} : i \in \mathcal{V})$ the binary signals emitted by all neurons at time t , where \mathcal{V} is the set of all neurons. Each neuron $i \in \mathcal{V}$ receives the signals emitted by a subset \mathcal{P}_i of neurons through directed links, known as *synapses*. Neurons in set \mathcal{P}_i are referred to as *presynaptic* for *postsynaptic* neuron i .

Membrane potential and filtered traces

The internal, analog state of each spiking neuron $i \in \mathcal{V}$ at time t is defined by its membrane potential $u_{i,t}$ (and possibly also by other secondary variables to be discussed) [15]. The value of the membrane potential indicates the probability of neuron i to spike. As illustrated in Figure 4, the membrane potential is the sum of the contributions from the incoming spikes of the presynaptic neurons and from the past spiking behavior of the neuron itself, where both contributions are filtered by the respective kernels a_t and b_t . To elaborate, we denote as $\mathbf{s}_{i,\leq t} = (s_{i,0}, \dots, s_{i,t})$ the spike signal emitted by neuron i up to time t . Given past input spike signals from the presynaptic neurons \mathcal{P}_i , denoted as $\mathbf{s}_{\mathcal{P}_i,\leq t-1} = \{\mathbf{s}_{j,\leq t-1}\}_{j \in \mathcal{P}_i}$, and the local spiking history $\mathbf{s}_{i,\leq t-1}$, the membrane potential of postsynaptic neuron i at time t can be written as [15]

$$u_{i,t} = \sum_{j \in \mathcal{P}_i} w_{j,i} \tilde{s}_{j,t-1} + w_{i,i} \tilde{s}_{i,t-1} + \gamma_i, \quad (1)$$

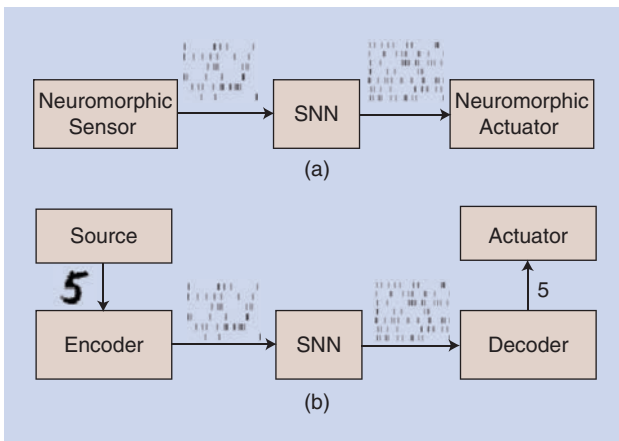


FIGURE 2. Depictions of the input/output interfaces of an SNN: (a) a direct interface with a neuromorphic sensor and actuator and (b) an indirect interface through encoding and decoding.

where the quantities $w_{j,i}$ for $j \in \mathcal{P}_i$ are synaptic (feedforward) weights, w_i is a feedback weight, γ_i is a bias parameter, and the quantities

$$\tilde{s}_{i,t} = a_t * s_{i,t} \text{ and } \bar{s}_{i,t} = b_t * s_{i,t} \quad (2)$$

are known as *filtered feedforward* and *feedback traces* of neuron i , respectively, where $*$ denotes the convolution operator $f_t * g_t = \sum_{\delta \geq 0} f_{t-\delta} g_{t-\delta}$.

Kernels and model weights

In (1) and (2), the filter a_t defines the synaptic response to a spike from a presynaptic neuron at the postsynaptic neuron. This filter is known as the *feedforward*, or *synaptic*, *kernel*. The filtered contribution of a spike from the presynaptic neuron $j \in \mathcal{P}_i$ is multiplied by a learnable weight $w_{j,i}$ for the synapse from neuron j to neuron $i \in \mathcal{V}$. When the filter is of finite duration τ , computing the feedforward trace $\tilde{s}_{i,t}$ requires keeping track of the window $\{s_{i,t}, s_{i,t-1}, \dots, s_{i,t-(\tau-1)}\}$ of prior synaptic inputs as part of the neuron's state [16]. An example is given by the “alpha” function $a_t = (\exp(-t/\tau_1) - \exp(-t/\tau_2))$ for $t = 0, \dots, \tau - 1$ and zero otherwise, with time constants τ_1 and τ_2 and duration τ , as illustrated in Figure 5(a). When the kernel is chosen as an infinitely long decaying exponential, i.e., as $a_t = \exp(-t/\tau_1)$, the feedforward trace $\tilde{s}_{i,t}$ can be directly computed using an autoregressive update that requires the storage of only a single scalar variable in the neuron's state [16], i.e., $\tilde{s}_{i,t} = \exp(-1/\tau_1)(\tilde{s}_{i,t-1} + s_{i,t})$. In general, the time constants and kernel shapes determine the synaptic memory and synaptic delays.

The filter b_t describes the response of a neuron to a local output spike and is known as a *feedback kernel*. A negative feedback kernel, such as $b_t = -\exp(-t/\tau_m)$, with time constant τ_m [see Figure 5(b)], models the refractory period upon the emission of a spike, with the time constant of the feedback kernel determining the duration of the refractory period.

As per (1), the filtered contribution of a local output spike is weighted by a learnable parameter w_i . Similar considerations as for the feedforward traces apply regarding the computation of the feedback trace.

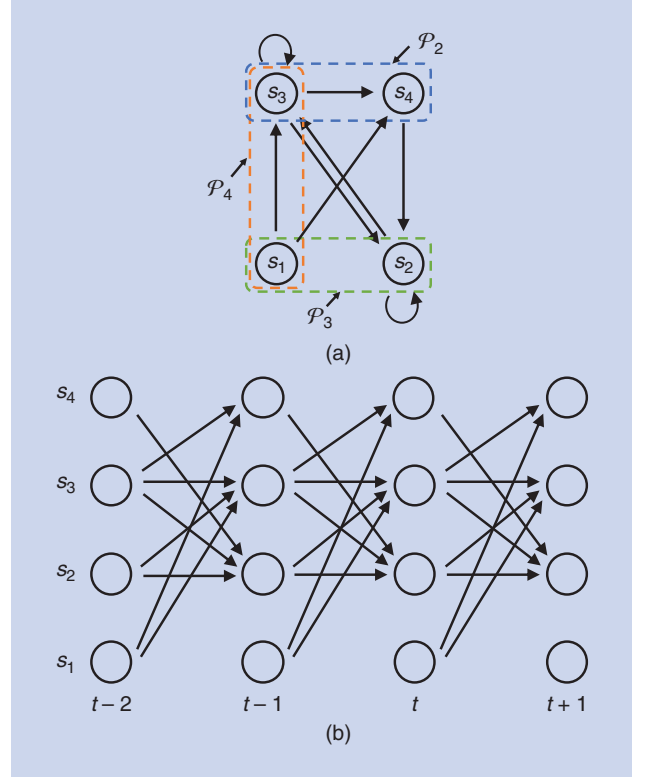


FIGURE 3. (a) An architecture of an SNN with $N = 4$ spiking neurons. The directed links between two neurons represent causal feedforward, or synaptic, dependencies, while the self-loop links represent feedback dependencies. The directed graph may have loops, including self-loops, indicating recurrent behavior. (b) A time-expanded view of the temporal dependencies implied by (a) with synaptic and feedback memories equal to one time step.

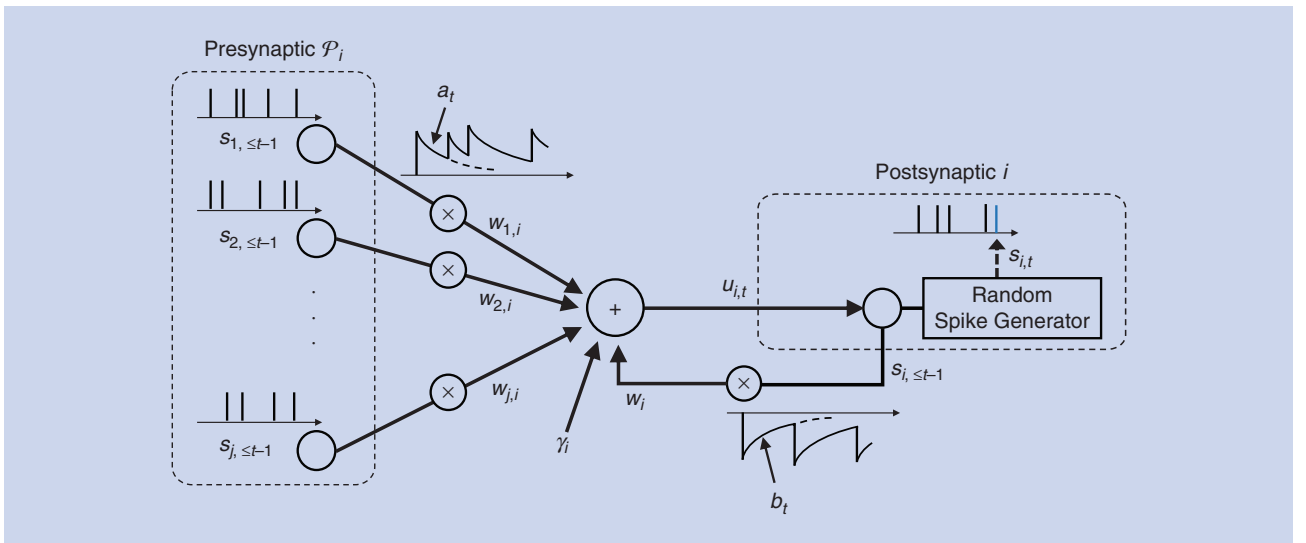


FIGURE 4. An illustration of the membrane potential model, with exponential feedforward and feedback kernels (see also Figure 5).

Generalizing the model described previously, a synapse can be associated with K_a learnable synaptic weights $\{w_{j,i,k}\}_{k=1}^{K_a}$. In this case, the contribution from presynaptic neuron j in (1) can be written as [14]

$$\left(\sum_{k=1}^{K_a} w_{j,i,k} a_{k,t} \right) * s_{j,t}, \quad (3)$$

where we have defined K_a fixed basis functions $\{a_{k,t}\}_{k=1}^{K_a}$, with learnable weights $\{w_{j,i,k}\}_{k=1}^{K_a}$. The feedback kernel can be similarly parameterized as the weighted sum of fixed K_b basis functions. Parameterization (3) makes it possible to adapt the shape of the filter applied by the synapse by learning the weights $\{w_{j,i,k}\}_{k=1}^{K_a}$. Typical examples of basis functions are the raised cosine functions shown in Figure 5(c). With this choice, the system can learn the sensitivity of each synapse to different synaptic delays, each corresponding to a different basis function, by adapting the weights $\{w_{j,i,k}\}_{k=1}^{K_a}$. In the rest of this article, with the exception of the “Batch Learning Examples” and “Online Learning Examples” sections, we focus on the simpler model of (1) and (2).

Practical implementations of the membrane potential model (1) can leverage the fact that linear filtering of binary spiking signals requires only carrying out sums while doing away with the need to compute expensive floating-point multiplications [5].

GLM

As discussed, a probabilistic model defines the joint probability distribution of the spike signals emitted by all neurons. In general, with the notation $s_{\leq t} = (s_0, \dots, s_t)$ using the chain rule, the log probability of the spike signals $s_{\leq T} = (s_0, \dots, s_T)$ emitted by all neurons in the SNN up to time T can be written as

$$\begin{aligned} \log p_{\theta}(s_{\leq T}) &= \sum_{t=0}^T \log p_{\theta}(s_t | s_{\leq t-1}) \\ &= \sum_{t=0}^T \sum_{i \in \mathcal{V}} \log p_{\theta_i}(s_{i,t} | s_{\mathcal{P}_i \cup \{i\}, \leq t-1}), \end{aligned} \quad (4)$$

where $\theta = \{\theta_i\}_{i \in \mathcal{V}}$ is the learnable parameter vector, with θ_i being the local parameters of neuron i . The decomposition (4)

is in terms of the conditional probabilities $p_{\theta_i}(s_{i,t} | s_{\mathcal{P}_i \cup \{i\}, \leq t-1})$, which represent the spiking probability of neuron i at time t , given its past spike timings and the past behaviors of its presynaptic neurons \mathcal{P}_i .

Under the GLM, the dependency of the spiking behavior of neuron $i \in \mathcal{V}$ on the history $s_{\mathcal{P}_i \cup \{i\}, \leq t-1}$ is mediated by the neuron’s membrane potential $u_{i,t}$. Specifically, the instantaneous firing probability of neuron i at time t is equal to

$$p_{\theta_i}(s_{i,t} = 1 | s_{\mathcal{P}_i \cup \{i\}, \leq t-1}) = p(s_{i,t} = 1 | u_{i,t}) = \sigma(u_{i,t}), \quad (5)$$

with $\sigma(\cdot)$ being the sigmoid function, i.e., $\sigma(x) = 1/(1 + \exp(-x))$. According to (5), a larger potential $u_{i,t}$ increases the probability that neuron i spikes. The model (5) is parameterized by the local learnable vector $\theta_i = \{\gamma_i, \{w_{j,i}\}_{j \in \mathcal{P}_i}, w_i\}$ of neuron i . SNNs modeled according to the described GLM framework can be thought of as a generalization of dynamic models of belief networks [17], and they can also be interpreted as a discrete-time version of Hawkes processes [18].

In a variant of this model, probability (5) can be written as $\sigma(u_{i,t}/\Delta u)$, where Δu is a bandwidth parameter that dictates the smoothness of the firing rate about the threshold. When taking the limit $\Delta u \rightarrow 0$, we obtain the deterministic integrate-and-fire model [19].

Relationship with ANNs

Under rate encoding, as long as the duration T is large enough, the deterministic integrate-and-fire model can mimic the operation of a conventional feedforward ANN with a nonnegative activation function. To this end, consider an ANN with an arbitrary topology defined by an acyclic directed graph. The corresponding SNN has the same topology, a feedforward kernel defined by a single basis function implementing a perfect integrator (i.e., a filter with a constant impulse response), the same synaptic weights of the ANN, and no feedback kernel. In this way, the value of the filtered feedforward trace for each synapse approximates the spiking rate of the presynaptic neuron as T increases. The challenge in enabling a conversion from ANN to SNN is to choose the thresholds γ_i and possibly a renormalization of the weights, so that the spiking rates of all neurons in the SNN approximate the outputs of the neurons in

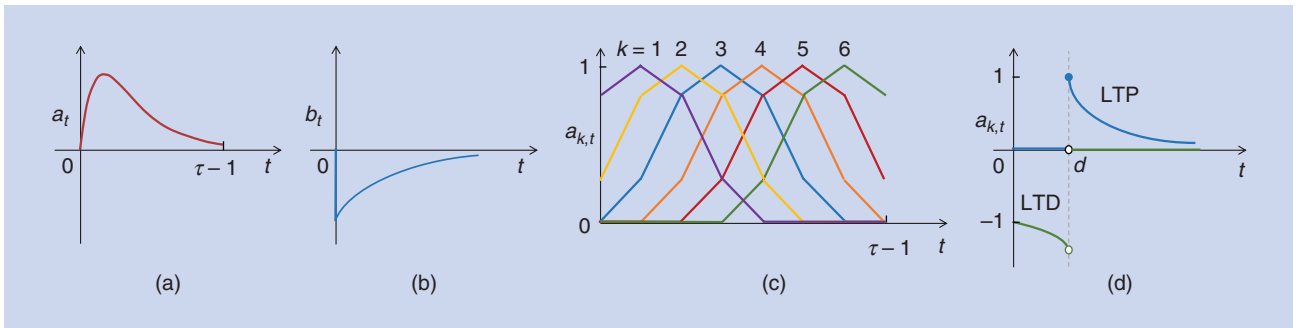


FIGURE 5. Examples of feedforward/feedback kernels: (a) an exponentially decaying feedforward kernel a_t , (b) an exponentially decaying feedback kernel b_t , (c) raised cosine basis functions $a_{k,t}$ in [14], and (d) spike-timing-dependent plasticity basis functions $a_{k,t}$ for long-term potentiation (LTP) and long-term depression (LTD), where the synaptic conduction delay equals d [16].

the ANN [6]. When including loops, deterministic SNNs can also implement recurrent NNs [20].

Gradient of the log-likelihood

The gradient of the log probability, or log-likelihood, $\mathcal{L}_{s \leq T}(\theta) = \log p_{\theta}(s \leq T)$ in (4), with respect to the learnable parameters θ , plays a key role in the problem of training a probabilistic SNN. Focusing on any neuron $i \in \mathcal{V}$, from (1) to (5), the gradient of the log-likelihood with respect to the local parameters θ_i for neuron i is given as

$$\nabla_{\theta_i} \mathcal{L}_{s \leq T}(\theta) = \sum_{t=0}^T \nabla_{\theta_i} \log p_{\theta_i}(s_{i,t} | s_{\mathcal{P}_i \cup \{i\}, \leq t-1}), \quad (6)$$

where the individual entries of the gradient of time t can be obtained as

$$\nabla_{\gamma_i} \log p_{\theta_i}(s_{i,t} | s_{\mathcal{P}_i \cup \{i\}, \leq t-1}) = s_{i,t} - \sigma(u_{i,t}), \quad (7a)$$

$$\nabla_{w_{ji}} \log p_{\theta_i}(s_{i,t} | s_{\mathcal{P}_i \cup \{i\}, \leq t-1}) = \bar{s}_{j,t-1}(s_{i,t} - \sigma(u_{i,t})), \quad (7b)$$

and

$$\nabla_{w_i} \log p_{\theta_i}(s_{i,t} | s_{\mathcal{P}_i \cup \{i\}, \leq t-1}) = \bar{s}_{i,t-1}(s_{i,t} - \sigma(u_{i,t})). \quad (7c)$$

The gradients (7) depend on the difference between the desired spiking behavior and its average behavior under the model distribution (5). The implications of this result for learning will be discussed in the next sections.

Training SNNs

SNNs can be trained using supervised, unsupervised, and reinforcement learning. To this end, the network follows a learning rule, which defines how the model parameters θ are updated on the basis of the available observations. As we will detail, learning rules can be applied in a batch mode at the end of a full period T of use of the SNN, based on multiple observations of duration T , or in an online fashion, i.e., after each time instant t , based on an arbitrarily long observation.

Locality

A learning rule is local if its operation can be decomposed into atomic steps that can be carried out in parallel at distributed processors based only on locally available information and limited communication on the connectivity graph (see Figure 3). Local information at a neuron includes the membrane potential, the feedforward filtered traces for the incoming synapses, the local feedback filtered trace, and the local model parameters. The processors will be considered here to be conventionally implemented at the level of individual neurons. Besides local signals, learning rules may also require global feedback signals, as discussed next.

Three-factor rule

While the details differ for each learning rule and task, a general form of the learning rule for the synaptic weights fol-

lows the three-factor rule [21], [22]. Accordingly, the synaptic weight $w_{j,i}$ from presynaptic neuron $j \in \mathcal{P}_i$ to a postsynaptic neuron $i \in \mathcal{V}$ is updated as

$$w_{j,i} \leftarrow w_{j,i} + \eta \times \ell \times \text{pre}_j \times \text{post}_i, \quad (8)$$

where η is a learning rate, ℓ is a scalar global learning signal that determines the sign and magnitude of the update, pre_j is a function of the activity of the presynaptic neuron $j \in \mathcal{P}_i$, and post_i depends on the activity of the postsynaptic neuron $i \in \mathcal{V}$. For most learning rules, pre- and postsynaptic terms are local to each neuron, while the learning signal ℓ , if present, plays the role of a global feedback signal. As a special case, the rule (8) can implement Hebb's hypothesis that "neurons that spike together wire together." This is indeed the case if the product of the pre_j and post_i terms is large when the two neurons spike at nearly the same time, resulting in a large change of the synaptic weight $w_{j,i}$ [10].

In the next two sections, we will see how learning rules of the form (8) can be derived in a principled manner as SGD updates obtained under the described probabilistic SNN models.

Training SNNs: Fully observed models

Fully observed versus partially observed models

Neurons in an SNN can be divided into the subsets of visible, or observed, neurons, which encode inputs and outputs, and hidden, or latent, neurons, whose role is to facilitate the desired behavior of the SNN. During training, the behavior of visible neurons is specified by the training data. For example, under supervised learning, input neurons are clamped to the input data, while the spiking signals of output neurons are determined by the desired output. Another related example is a reinforcement learning task in which the SNN models a policy, with input neurons encoding the state and output neurons encoding the action previously taken by the learner in response to the given input [23].

In the case of fully observed models, the SNN contains only visible neurons while, in the case of partially observed models, the SNN also includes hidden neurons. We first consider the simpler former case and then extend the discussion to partially observed models.

Maximum likelihood learning via SGD

The standard training criterion for probabilistic models for both supervised and unsupervised learning is maximum likelihood (ML). ML selects model parameters that maximize the probability of the observed data and, hence, of the desired input/output behavior under the model. To elaborate, we consider an example $\mathbf{x}_{\leq T}$ consisting of fully observed spike signals for all neurons in the SNN, including both input and output neurons. Using the notation in the "Models" section, we hence have $\mathbf{s}_{\leq T} = \mathbf{x}_{\leq T}$. During training, the spike signals for all neurons are thus clamped to the values assumed in the data point $\mathbf{x}_{\leq T}$, and the log-likelihood is given as $\mathcal{L}_{\mathbf{x}_{\leq T}}(\theta) = \log p_{\theta}(\mathbf{x}_{\leq T})$

in (4), with $s_{\leq T} = \mathbf{x}_{\leq T}$. As we will see next, for batch learning, there are multiple such examples $\mathbf{x}_{\leq T}$ in the training set while, for online learning, we have a single arbitrary long example $\mathbf{x}_{\leq T}$ for large T .

Batch SGD

In the batch training mode, a training set $\mathcal{D} = \{\mathbf{x}_{\leq T}^m\}_{m=1}^M$ of M fully observed examples is available to enable the learning of the model parameters. The batch SGD-based rule proceeds iteratively by selecting an example $\mathbf{x}_{\leq T}$ from the training set \mathcal{D} at each iteration (see, e.g., [24]). The model parameters θ are then updated in the direction of the gradient (6) and (7), with $s_{\leq T} = \mathbf{x}_{\leq T}$, as

$$\theta \leftarrow \theta + \eta \nabla_{\theta} \mathcal{L}_{\mathbf{x}_{\leq T}}(\theta), \quad (9)$$

where the learning rate η is assumed to be fixed here for simplicity of notation. Note that the update (9) is applied at the end of the observation period T . The batch algorithm can be generalized by summing over a minibatch of examples at each iteration [24].

Online SGD

In the online training mode, an arbitrary long example $\mathbf{x}_{\leq T}$ is available, and the model parameters θ are updated at each time t (or, more generally, periodically every few time instants). This can be done by introducing an eligibility trace $\mathbf{e}_{i,t}$ for each neuron i [19], [22]. As summarized in Algorithm 1, the eligibility trace $\mathbf{e}_{i,t}$ in (A1), with $\kappa < 1$, computes a weighted average of current and past gradient updates. In this update, the current gradient $\nabla_{\theta} \log p_{\theta}(\mathbf{x}_t | \mathbf{x}_{\leq t-1})$ is weighted by a factor $(1 - \kappa)$, and the gradient that is evaluated l steps in the past is multiplied by the exponentially decaying coefficient $(1 - \kappa) \cdot \kappa^l$. The eligibility trace captures the impact of past updates on the current spiking behavior, and it can help stabilize the online training by reducing the variance of the updates (for sufficiently large κ) [13].

Algorithm 1. ML training via online SGD.

Input: Training example $\mathbf{x}_{\leq T}$ and learning rates η and κ
Output: Learned model parameters θ

```

1: initialize parameters  $\theta$ 
2: repeat
3:   for each  $t = 0, 1, \dots, T$ 
4:     for each neuron  $i \in \mathcal{V}$  do
5:       compute the gradient  $\nabla_{\theta_i} \log p_{\theta_i}(\mathbf{x}_{i,t} | \mathbf{x}_{\mathcal{P}_i \cup \{i\}, \leq t-1})$  with respect
         to the local parameters  $\theta_i$  from (7)
6:       compute the eligibility trace  $\mathbf{e}_{i,t}$ 
            $\mathbf{e}_{i,t} = \kappa \mathbf{e}_{i,t-1} + (1 - \kappa) \nabla_{\theta_i} \log p_{\theta_i}(\mathbf{x}_{i,t} | \mathbf{x}_{\mathcal{P}_i \cup \{i\}, \leq t-1})$  (A1)
7:       update the local model parameters
            $\theta_i \leftarrow \theta_i + \eta \mathbf{e}_{i,t}$  (A2)
8:     end
9:   until stopping criterion is satisfied.
```

Interpretation

The online gradient update for any synaptic weight $w_{j,i}$ can be interpreted in light of the general form of rule (8). In fact, the gradient (7b) has a two-factor form, whereby the global learning signal is absent; the presynaptic term is given by the filtered feedforward trace $\tilde{x}_{j,t-1}$ of the presynaptic neuron $j \in \mathcal{P}_i$, and the postsynaptic term is given by the error term $x_{i,t} - \sigma(u_{i,t})$. This error measures the difference between the desired spiking behavior of the postsynaptic neuron i at any time t and its average behavior under the model distribution (5).

This update can be related to the standard spike-timing-dependent plasticity (STDP) rule [10], [16], [25]. In fact, STDP stipulates that the long-term potentiation (LTP) of a synapse occurs when the presynaptic neuron spikes right before a postsynaptic neuron, while long-term depression (LTD) of a synapse takes place when the presynaptic neuron spikes right after a postsynaptic neuron. With the basis functions depicted in Figure 5(d), if a presynaptic spike occurs more than d steps prior to the postsynaptic spike at time t , an increase in the synaptic weight, or LTP, occurs, while a decrease in the synaptic weight, or LTD, takes place otherwise [16]. The parameter d can hence be interpreted as synaptic delay.

As for the synaptic weights, all other gradients (7) also depend on an error signal measuring the gap between the desired and average model behavior. In (7a)–(7c), the desired behavior is given by samples $s_{i,t} = x_{i,t}$ in the training example. The contribution of this error signal can be interpreted as a form of (task-specific) homeostatic plasticity, in that it regulates the neuronal firing rates around desirable set-point values [10], [26].

Locality and implementation

Given the absence of a global learning signal, the online SGD rule in Algorithm 1 and the batch SGD rule can be implemented locally, so that each neuron i updates its own local parameters θ_i . Each neuron i uses information about the local spike signal $x_{i,t}$, the feedforward filtered traces $\tilde{x}_{j,t-1}$ for all presynaptic neurons $j \in \mathcal{P}_i$, and the local feedback filtered trace $\tilde{x}_{i,t-1}$ to compute the first terms in (7a)–(7c), while the second terms in (7a)–(7c) are obtained from (5) by using the neuron's membrane potential $u_{i,t}$.

Training SNNs: Partially observed models

Latent neurons

As mentioned previously, the set \mathcal{V} of neurons can be partitioned into the disjoint subsets of observed (input and output) and hidden neurons. The N_X neurons in the subset \mathcal{X} are observed, and the N_H neurons in the subset \mathcal{H} are hidden, or latent, and we have $\mathcal{V} = \mathcal{X} \cup \mathcal{H}$. We write as $\mathbf{x}_t = (x_{i,t} : i \in \mathcal{X})$ and $\mathbf{h}_t = (h_{i,t} : i \in \mathcal{H})$, the binary signals emitted by the observed and hidden neurons at time t , respectively. Therefore, using the notation in the “Models” section, we have $s_{i,t} = x_{i,t}$ for any observed neuron $i \in \mathcal{X}$ and $s_{i,t} = h_{i,t}$ for any latent neuron $i \in \mathcal{H}$ as well as $\mathbf{s}_t = (\mathbf{x}_t, \mathbf{h}_t)$ for the overall set of spike signals at time t . During training, the spike signals $\mathbf{x}_{\leq T}$ of the

observed neurons are clamped to the examples in the training set while the probability distribution of the signals $\mathbf{h}_{\leq T}$ of the hidden neurons can be adapted to ensure the desired input/output behavior. Mathematically, the probabilistic model is defined as in (4) and (5), with $s \leq T = (\mathbf{x}_{\leq T}, \mathbf{h}_{\leq T})$.

ML via SGD and variational learning

Here, we review a standard learning rule that tackles the ML problem by using SGD. Unlike in the fully observed case, as we will see, variational inference is needed to cope with the complexity of computing the gradient of the log-likelihood of the observed spike signals in the presence of hidden neurons [12].

Log-likelihood

The log-likelihood of an example of observed spike signals $\mathbf{x}_{\leq T}$ is obtained via marginalization by summing over all possible values of the latent spike signals $\mathbf{h}_{\leq T}$ as $\mathcal{L}_{\mathbf{x}_{\leq T}}(\boldsymbol{\theta}) = \log p_{\boldsymbol{\theta}}(\mathbf{x}_{\leq T}) = \log \sum_{\mathbf{h}_{\leq T}} p_{\boldsymbol{\theta}}(\mathbf{x}_{\leq T}, \mathbf{h}_{\leq T})$. Let us denote as $\langle \cdot \rangle_p$ the expectation over a distribution p , as in $\langle f(x) \rangle_{p(x)} = \sum_x f(x)p(x)$, for some function $f(x)$. The gradient of the log-likelihood with respect to the model parameters $\boldsymbol{\theta}$ can be expressed as (see, e.g., [12, Ch. 6])

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathbf{x}_{\leq T}}(\boldsymbol{\theta}) = \langle \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}_{\leq T}, \mathbf{h}_{\leq T}) \rangle_{p_{\boldsymbol{\theta}}(\mathbf{h}_{\leq T} | \mathbf{x}_{\leq T})}, \quad (10)$$

where the expectation is with respect to the posterior distribution $p_{\boldsymbol{\theta}}(\mathbf{h}_{\leq T} | \mathbf{x}_{\leq T})$ of the latent variables $\mathbf{h}_{\leq T}$, given the observation $\mathbf{x}_{\leq T}$. Note that the gradient $\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}_{\leq T}, \mathbf{h}_{\leq T}) = \sum_{t=0}^T \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}_t, \mathbf{h}_t | \mathbf{x}_{\leq t-1}, \mathbf{h}_{\leq t-1})$ is obtained from (7), with $s \leq T = (\mathbf{x}_{\leq T}, \mathbf{h}_{\leq T})$. Computing the posterior $p_{\boldsymbol{\theta}}(\mathbf{h}_{\leq T} | \mathbf{x}_{\leq T})$ amounts to the Bayesian inference of the hidden spike signals for the observed values $\mathbf{x}_{\leq T}$. Given that we have the equality $p_{\boldsymbol{\theta}}(\mathbf{h}_{\leq T} | \mathbf{x}_{\leq T}) = p_{\boldsymbol{\theta}}(\mathbf{x}_{\leq T}, \mathbf{h}_{\leq T}) / p_{\boldsymbol{\theta}}(\mathbf{x}_{\leq T})$, this task requires the evaluation of the marginal distribution $p_{\boldsymbol{\theta}}(\mathbf{x}_{\leq T}) = \sum_{\mathbf{h}_{\leq T}} p_{\boldsymbol{\theta}}(\mathbf{x}_{\leq T}, \mathbf{h}_{\leq T})$. For problems of practical size, this computation is intractable, and, hence, so is evaluating the gradient (10).

Variational learning

Variational inference, or variational Bayes, approximates the true posterior distribution $p_{\boldsymbol{\theta}}(\mathbf{h}_{\leq T} | \mathbf{x}_{\leq T})$ by means of any arbitrary variational posterior distribution $q_{\phi}(\mathbf{h}_{\leq T} | \mathbf{x}_{\leq T})$ parameterized by a vector ϕ of learnable parameters. For any variational distribution $q_{\phi}(\mathbf{h}_{\leq T} | \mathbf{x}_{\leq T})$, using Jensen's inequality, the log-likelihood $\mathcal{L}_{\mathbf{x}_{\leq T}}(\boldsymbol{\theta})$ can be lower-bounded as (see, e.g., [12, Ch. 6 and Ch. 8])

$$\begin{aligned} \mathcal{L}_{\mathbf{x}_{\leq T}}(\boldsymbol{\theta}) &= \log \sum_{\mathbf{h}_{\leq T}} p_{\boldsymbol{\theta}}(\mathbf{x}_{\leq T}, \mathbf{h}_{\leq T}) \\ &\geq \sum_{\mathbf{h}_{\leq T}} q_{\phi}(\mathbf{h}_{\leq T} | \mathbf{x}_{\leq T}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_{\leq T}, \mathbf{h}_{\leq T})}{q_{\phi}(\mathbf{h}_{\leq T} | \mathbf{x}_{\leq T})} \\ &= \langle \ell_{\boldsymbol{\theta}, \phi}(\mathbf{x}_{\leq T}, \mathbf{h}_{\leq T}) \rangle_{q_{\phi}(\mathbf{h}_{\leq T} | \mathbf{x}_{\leq T})} := L_{\mathbf{x}_{\leq T}}(\boldsymbol{\theta}, \phi), \end{aligned} \quad (11)$$

where we have defined the learning signal as

$$\ell_{\boldsymbol{\theta}, \phi}(\mathbf{x}_{\leq T}, \mathbf{h}_{\leq T}) := \log p_{\boldsymbol{\theta}}(\mathbf{x}_{\leq T}, \mathbf{h}_{\leq T}) - \log q_{\phi}(\mathbf{h}_{\leq T} | \mathbf{x}_{\leq T}). \quad (12)$$

A baseline variational learning rule, also known as the *variational expectation maximization* algorithm, is based on the maximization of the evidence lower bound (ELBO) $L_{\mathbf{x}_{\leq T}}(\boldsymbol{\theta}, \phi)$ in (11) with respect to both the model parameters $\boldsymbol{\theta}$ and the variational parameters ϕ . Accordingly, for a given observed example $\mathbf{x}_{\leq T} \in \mathcal{D}$, the learning rule is given by gradient ascent updates, where the gradients can be computed as

$$\nabla_{\boldsymbol{\theta}} L_{\mathbf{x}_{\leq T}}(\boldsymbol{\theta}, \phi) = \langle \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}_{\leq T}, \mathbf{h}_{\leq T}) \rangle_{q_{\phi}(\mathbf{h}_{\leq T} | \mathbf{x}_{\leq T})}, \quad (13a)$$

and

$$\begin{aligned} \nabla_{\phi} L_{\mathbf{x}_{\leq T}}(\boldsymbol{\theta}, \phi) &= \\ \langle \ell_{\boldsymbol{\theta}, \phi}(\mathbf{x}_{\leq T}, \mathbf{h}_{\leq T}) \cdot \nabla_{\phi} \log q_{\phi}(\mathbf{h}_{\leq T} | \mathbf{x}_{\leq T}) \rangle_{q_{\phi}(\mathbf{h}_{\leq T} | \mathbf{x}_{\leq T})}, \end{aligned} \quad (13b)$$

respectively. The gradient (13a) is derived in a manner analogous to (10), and the gradient (13b) is obtained from the standard REINFORCE, or score function, gradient [12, Ch. 8], [27]. Importantly, the gradients (13) require expectations with respect to the known variational posterior $q_{\phi}(\mathbf{h}_{\leq T} | \mathbf{x}_{\leq T})$ evaluated at the current value of variational parameters ϕ rather than with respect to the hard-to-compute posterior $p_{\boldsymbol{\theta}}(\mathbf{h}_{\leq T} | \mathbf{x}_{\leq T})$. An alternative to the computation of the gradient over the variational parameters ϕ as in (13b) is given by the so-called reparameterization trick [28], as briefly discussed in the ‘‘Conclusions and Open Problems’’ section.

In practice, computing the averages in (13) is still intractable because of the large domain of the hidden variables $\mathbf{h}_{\leq T}$. Therefore, the expectations over the variational posterior are typically approximated by means of Monte Carlo empirical averages. This is possible as long as sampling from the variational posterior $q_{\phi}(\mathbf{h}_{\leq T} | \mathbf{x}_{\leq T})$ is feasible. As an example, if a single spike signal $\mathbf{h}_{\leq T}$ is sampled from $q_{\phi}(\mathbf{h}_{\leq T} | \mathbf{x}_{\leq T})$, we obtain the Monte Carlo approximations of (13) as

$$\nabla_{\boldsymbol{\theta}} \hat{L}_{\mathbf{x}_{\leq T}}(\boldsymbol{\theta}, \phi) = \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}_{\leq T}, \mathbf{h}_{\leq T}), \quad (14a)$$

and

$$\nabla_{\phi} \hat{L}_{\mathbf{x}_{\leq T}}(\boldsymbol{\theta}, \phi) = \ell_{\boldsymbol{\theta}, \phi}(\mathbf{x}_{\leq T}, \mathbf{h}_{\leq T}) \cdot \nabla_{\phi} \log q_{\phi}(\mathbf{h}_{\leq T} | \mathbf{x}_{\leq T}). \quad (14b)$$

Batch doubly SGD

In a batch training formulation, at each iteration, an example $\mathbf{x}_{\leq T}$ is selected from the training set \mathcal{D} . At the end of the observation period T , both model and variational parameters can be updated in the direction of the gradients $\nabla_{\boldsymbol{\theta}} \hat{L}_{\mathbf{x}_{\leq T}}(\boldsymbol{\theta}, \phi)$ and $\nabla_{\phi} \hat{L}_{\mathbf{x}_{\leq T}}(\boldsymbol{\theta}, \phi)$ in (14) as

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} \hat{L}_{\mathbf{x}_{\leq T}}(\boldsymbol{\theta}, \phi), \quad (15a)$$

and

$$\phi \leftarrow \phi + \eta_{\phi} \nabla_{\phi} \hat{L}_{\mathbf{x}_{\leq T}}(\boldsymbol{\theta}, \phi), \quad (15b)$$

respectively, where the learning rates $\eta_{\boldsymbol{\theta}}$ and η_{ϕ} are assumed to be fixed for simplicity. Rule (15) is known as *doubly SGD*

since sampling is carried out over both the observed examples $\mathbf{x}_{\leq T}$ in the training set and the hidden spike signals $\mathbf{h}_{\leq T}$.

The doubly stochastic gradient estimator (14b) typically exhibits a high variance. To reduce the variance, a common approach is to subtract a baseline control variate from the learning signal. This can be done by replacing the learning signal in (14b) with the centered learning signal $\ell_{\theta, \phi}(\mathbf{x}_{\leq T}, \mathbf{h}_{\leq T}) - \bar{\ell}$, where the baseline $\bar{\ell}$ is calculated as a moving average of learning signals computed at previous iterations [22], [27], [29].

Online doubly SGD

The batch doubly SGD rule (15) applies with any choice of variational distribution $q_{\phi}(\mathbf{h}_{\leq T}|\mathbf{x}_{\leq T})$, as long as it is feasible to sample from it and to compute the gradient in (14b). However, the locality properties and complexity of the learning rule are strongly dependent on the choice of the variational distribution. We now discuss a specific choice considered in [16], [22], and [29]–[31] that yields an online rule, summarized in Algorithm 2.

The approach approximates the true posterior $p_{\theta}(\mathbf{h}_{\leq T}|\mathbf{x}_{\leq T})$ with a feedforward distribution that ignores the stochastic dependence of the hidden spike signals \mathbf{h}_t at time t on the future values of the observed spike signals $\mathbf{x}_{\leq T}$. The corresponding variational distribution can be written as

$$q_{\theta^H}(\mathbf{h}_{\leq T}|\mathbf{x}_{\leq T}) = \prod_{t=0}^T p_{\theta^H}(\mathbf{h}_t|\mathbf{x}_{\leq t-1}, \mathbf{h}_{\leq t-1}) = \prod_{t=0}^T \prod_{i \in \mathcal{H}} p(h_{i,t}|u_{i,t}), \quad (16)$$

Algorithm 2. ML training via online doubly SGD.

Input: Training data $\mathbf{x}_{\leq T}$ and learning rates η and κ
Output: Learned model parameters θ

```

1: initialize parameters  $\theta$ 
2: repeat
3:   feedforward sampling:
4:   for each hidden neuron  $i \in \mathcal{H}$  do
5:     emit a spike  $h_{i,t} = 1$  with probability  $\sigma(u_{i,t})$ 
6:   end
7:   global feedback:
8:   a central processor collects the log probabilities  $p(x_{i,t}|u_{i,t})$  in
   (5) from all observed neurons  $i \in \mathcal{X}$ , computes an eligibility
   trace from the learning signal (17) as
       
$$\ell_t = \kappa \ell_{t-1} + (1 - \kappa) \sum_{i \in \mathcal{X}} \log p(x_{i,t}|u_{i,t}), \quad (A3)$$

   and feeds back the global learning signal  $\ell_t$  to all latent neurons
9:   parameter update:
10:  for each neuron  $i \in \mathcal{V}$  do
11:    evaluate the eligibility trace  $\mathbf{e}_{i,t}$  as
       
$$\mathbf{e}_{i,t} = \kappa \mathbf{e}_{i,t-1} + (1 - \kappa) \nabla_{\theta_i} \log p_{\theta_i}(s_{i,t}|\mathbf{x}_{\leq t-1}, \mathbf{h}_{\leq t-1}), \quad (A4)$$

   with  $s_{i,t} = x_{i,t}$  if  $i \in \mathcal{X}$  and  $s_{i,t} = h_{i,t}$  if  $i \in \mathcal{H}$ 
12:    update the local model parameters as
       
$$\theta_i \leftarrow \theta_i + \eta \cdot \begin{cases} \mathbf{e}_{i,t_r} & \text{if } i \in \mathcal{X} \\ \ell_t \mathbf{e}_{i,t_r} & \text{if } i \in \mathcal{H} \end{cases} \quad (A5)$$

13:  end
14: until stopping criterion is satisfied.
```

where we denote as $\theta^H = \{\theta_i\}_{i \in \mathcal{H}}$ the collection of the model parameters for hidden neurons, and $p(h_{i,t} = 1|u_{i,t}) = \sigma(u_{i,t})$ by (5), with $s_{i,t} = h_{i,t}$. We note that (16) is an approximation of the true posterior $p_{\theta}(\mathbf{h}_{\leq T}|\mathbf{x}_{\leq T}) = \prod_{t=0}^T p_{\theta}(\mathbf{h}_t|\mathbf{x}_{\leq T}, \mathbf{h}_{\leq t-1})$ since it neglects the correlation between variables \mathbf{h}_t and the future observed samples $\mathbf{x}_{\geq t}$. In (16), we have emphasized that the variational parameters ϕ are tied to a subset of the model parameters, as per the equality $\phi = \theta^H$. As a result, this choice of variational distribution does not include additional learnable parameters apart from the model parameters θ . The learning signal (12) with the feedforward distribution (16) reads

$$\begin{aligned} \ell_{\theta^X}(\mathbf{x}_{\leq T}, \mathbf{h}_{\leq T}) &= \sum_{t=0}^T \log p_{\theta^X}(\mathbf{x}_t|\mathbf{x}_{\leq t-1}, \mathbf{h}_{\leq t-1}) \\ &= \sum_{t=0}^T \sum_{i \in \mathcal{X}} \log p(x_{i,t}|u_{i,t}), \end{aligned} \quad (17)$$

where $\theta^X = \{\theta_i\}_{i \in \mathcal{X}}$ is the collection of the model parameters for observed neurons.

With the choice of (16) for the variational posterior, the batch doubly SGD update rule (15) can be turned into an online rule by generalizing Algorithm 1, as detailed in Algorithm 2. At each step of the online procedure, each hidden neuron $i \in \mathcal{H}$ emits a spike, i.e., $h_{i,t} = 1$, at any time t by following the current model distribution (16), i.e., with probability $\sigma(u_{i,t})$. Note that the membrane potential $u_{i,t}$ of any neuron i at time t is obtained from (1), with observed neurons clamped to the training example $\mathbf{x}_{\leq t-1}$ and hidden neurons clamped to the samples $\mathbf{h}_{\leq t-1}$. Then, a central processor collects the log probabilities $p(x_{i,t}|u_{i,t})$ under the current model from all observed neurons $i \in \mathcal{X}$ to compute the eligibility trace of the learning signal ℓ_t , as in (A3) and feeds back the global learning signal to all latent neurons.

Intuitively, this learning signal indicates to the hidden neurons how effective their current signaling is in ensuring the desired input/output behavior with high probability. Finally, each observed and hidden neuron i computes the eligibility trace $\mathbf{e}_{i,t}$ of the gradient, i.e., $\nabla_{\theta_i} \log p_{\theta_i}(x_{i,t}|\mathbf{x}_{\leq t-1}, \mathbf{h}_{\leq t-1})$ and $\nabla_{\theta_i} \log p_{\theta_i}(h_{i,t}|\mathbf{x}_{\leq t-1}, \mathbf{h}_{\leq t-1})$, respectively, as in (A4). The local parameters θ_i of each observed neuron $i \in \mathcal{X}$ are updated in the direction of the eligibility trace $\mathbf{e}_{i,t}$, while each hidden neuron $i \in \mathcal{H}$ updates the parameter using $\mathbf{e}_{i,t}$ and the learning signal ℓ_t in (A3).

Sparsity and regularization

As discussed, the energy consumption of SNNs depends on the number of spikes emitted by the neurons. Since the ML criterion does not enforce any sparsity constraint, an SNN trained using the methods discussed so far may present dense spiking signals [18]. This is especially the case for the hidden neurons, whose behavior is not tied to the training data. To obviate this problem, it is possible to add a regularization term $-\alpha \cdot \text{KL}(q_{\phi}(\mathbf{h}_{\leq T}|\mathbf{x}_{\leq T})\|r(\mathbf{h}_{\leq T}))$ to the learning objective $L_{\mathbf{x}_{\leq T}}(\theta, \phi)$ in (11), where $\text{KL}(p\|q) = \sum_x p(x) \log(p(x)/q(x))$ is the Kullback–Leibler divergence between distributions p and q , $r(\mathbf{h}_{\leq T})$ represents a baseline distribution with the desired

level of sparsity, and $\alpha > 0$ is a parameter adjusting the amount of regularization. This regularizing term, which penalizes variational distributions far from the baseline distribution, can also act as a regularizer to minimize overfitting by enforcing a bounded rationality constraint [32]. The learning rule in Algorithm 2 can be modified accordingly.

Interpretation

The update (A5) for the synaptic weight $w_{j,i}$ of any observed neuron $i \in \mathcal{X}$ follows the local two-factor rule, as described in the “Interpretation” section. In contrast, for any hidden neuron $i \in \mathcal{H}$, the update applies a three-factor nonlocal learning rule (8). Accordingly, the postsynaptic error signal of hidden neuron i and the filtered feedforward trace of presynaptic neuron j are multiplied by the global learning signal (17). As anticipated, the global learning signal can be interpreted as an internal reward signal. To see this more generally, we can rewrite (17) as

$$\ell_{\theta^*}(\mathbf{x}_{\leq T}, \mathbf{h}_{\leq T}) = \log p_{\theta}(\mathbf{x}_{\leq T} | \mathbf{h}_{\leq T}) - \log \frac{q_{\theta^*}(\mathbf{h}_{\leq T} | \mathbf{x}_{\leq T})}{p_{\theta}(\mathbf{h}_{\leq T})}. \quad (18)$$

According to (18), the learning signal rewards hidden spike signals $\mathbf{h}_{\leq T}$, producing observations $\mathbf{x}_{\leq T}$ that yield a large likelihood $\log p_{\theta}(\mathbf{x}_{\leq T} | \mathbf{h}_{\leq T})$ for the desired behavior. Furthermore, it penalizes values of hidden spike signals $\mathbf{h}_{\leq T}$ that have large variational probability $q_{\theta^*}(\mathbf{h}_{\leq T} | \mathbf{x}_{\leq T})$ while having a low prior probability $p_{\theta}(\mathbf{h}_{\leq T})$ under the model.

As discussed in the “Learning Tasks” section, SNNs can be trained in a batch or online mode. In the next sections, we provide a representative, simple, and reproducible example for each case.

Batch learning examples

As an example of batch learning, we consider the standard handwritten digit classification task on the USPS data set [35]. We adopt an SNN with two layers, the first encoding the input and the second the output, with directed synaptic links existing from all neurons in the input layer to all neurons in the output layer. No hidden neurons exist, and, hence, training can be done as described in the section “Training SNNs: Fully Observed Models.” Each 16×16 input image, representing either a one or a seven handwritten digit, is encoded in the spike domain by using rate encoding. Each gray pixel is converted into an input spiking signal by generating an independent identically distributed (i.i.d.) Bernoulli vector of T samples, with the spiking probability proportional to the pixel intensity and limited to between zero and 0.5. As a result, we have 256 input neurons, with one per pixel of the input image. The digit labels {1, 7} are also rate encoded using each one of the two output neurons. The neuron corresponding to the correct label index emits spikes with a frequency of one every three samples, while the other output neurons are silent. We refer the reader to [33] and the supplementary material [34] for further details on the numerical setup.

Figure 6 shows the classification accuracy in the test set versus the duration T of the operation of the SNN after the convergence of the training process. The classification accuracy of a conventional ANN with the same topology and a softmax output layer is added for comparison. Note that, unlike the SNN, the ANN outputs real values, namely, the logits for each class processed by the soft-max layer. From the figure, the SNN is seen to provide a graceful tradeoff between accuracy and complexity of learning: as T increases, the number of spikes that are processed and the output by the SNN grow larger, entailing a larger inference complexity but also an improved accuracy that tends to that of the baseline ANN.

Online learning examples

We now consider an online prediction task in which the SNN sequentially observes a time sequence $\{a_l\}$ and the SNN is trained to predict, in an online manner, the next value of sequence a_l , given the observation of the previous values $a_{\leq l-1}$. The time sequence $\{a_l\}$ is encoded in the spike domain, producing a spike signal $\{\mathbf{x}_l\}$, consisting of N_X spiking signals $\mathbf{x}_l = (x_{1,l}, \dots, x_{N_X,l})$ with $\Delta T \geq 1$ samples for each sample a_l . We refer to ΔT as a time expansion factor. Each of the spiking signals $x_{i,l}$ is associated with one of N_X visible neurons.

We adopt a fully connected SNN topology that also includes N_H hidden neurons. In this online prediction task, we trained the SNN using Algorithm 2, with the addition of a sparsity regularization term. This is obtained by assuming an i.i.d. reference Bernoulli distribution with a desired spiking rate $r \in [0, 1]$, i.e., $\log r(\mathbf{h}_{\leq T}) = \sum_{i=0}^T \sum_{i \in \mathcal{H}} h_{i,t} \log r + (1 - h_{i,t}) \log(1 - r)$ (see the supplementary material [34] for details). The source sequence is randomly generated as follows: at every $T_s = 25$ time steps, one of three possible sequences of duration T_s is selected, namely, an all-zero sequence with probability 0.7, a sequence of class 1 from the SwedishLeaf data set of the UCR archive [36], or a sequence of class 6 from the same archive, with equal probability [see Figure 7(a) for an illustration].

Encoding and decoding

Each value a_l of the time sequence is converted into ΔT samples $\mathbf{x}_{l\Delta T+1}, \mathbf{x}_{l\Delta T+2}, \dots, \mathbf{x}_{(l+1)\Delta T}$ of the N_X spike signals $\{\mathbf{x}_l\}$

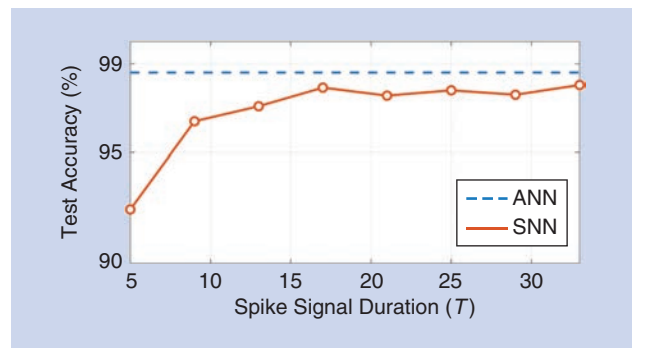


FIGURE 6. Performance of classification based on a two-layer SNN trained via batch ML learning in terms of accuracy versus the duration T of the operation of the SNN. The accuracy of an ANN with the same topology is also shown as a baseline (see [33] and [34] for details).

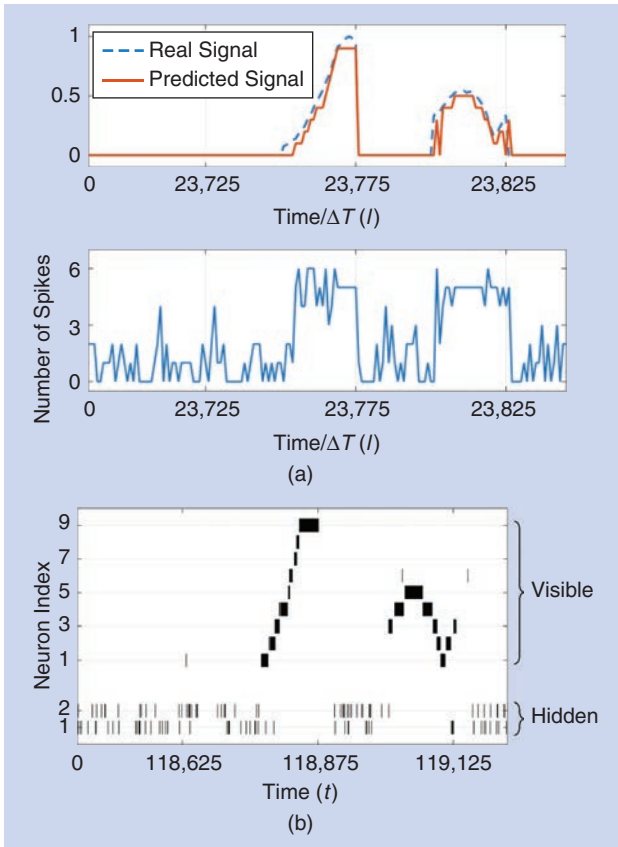


FIGURE 7. An online prediction task based on an SNN with $N_X = 9$ visible neurons and $N_H = 2$ hidden neurons trained via Algorithm 2. (a) A real analog time signal and a predicted decoded signal (top), and the total number of spikes emitted by the SNN (bottom). (b) A spike raster plot of visible neurons (top) and a spike raster plot of hidden neurons (bottom).

via rate or time coding, as illustrated in Figure 8. With rate coding, the value a_l is first discretized into $N_X + 1$ uniform quantization levels using rounding to the largest lower value. The lowest, silent level is converted to all-zero signals $\mathbf{x}_{l\Delta T+1}, \mathbf{x}_{l\Delta T+2}, \dots, \mathbf{x}_{(l+1)\Delta T}$. Each of the other N_X levels is assigned to a visible neuron, so that the neuron associated with the quantization level corresponding to value a_l emits ΔT consecutive spikes while the other neurons are silent. Rate decoding predicts value a_{l+1} by generating the samples $\mathbf{x}_{(l+1)\Delta T+1}, \dots, \mathbf{x}_{(l+2)\Delta T}$ from the trained model and then selecting the neuron with the largest number of spikes in this window.

For time coding, each of the N_X visible neurons is associated with a different shifted, truncated Gaussian receptive field [37]. Accordingly, as seen in Figure 8(b), for each value a_l , each visible neuron i emits a signal $x_{i,l\Delta T+1}, x_{i,l\Delta T+2}, \dots, x_{i,(l+1)\Delta T}$ that contains no spike if the value a_l is outside the receptive field and, otherwise, contains one spike, with the timing determined by the value of the corresponding truncated Gaussian receptive field quantized to values $\{1, \dots, \Delta T\}$ using rounding to the nearest value. Time decoding considers the first spike timing of the samples $x_{i,(l+1)\Delta T+1}, \dots, x_{i,(l+2)\Delta T}$ for each visible neuron i and predicts a value a_{l+1} using a least-squares criterion on the values of the receptive fields (see [11] and [37]). We refer to the supplementary material [34] for further details on the numerical setup.

Rate coding

First, assuming rate encoding with $\Delta T = 5$, we train an SNN with $N_X = 9$ visible neurons and $N_H = 2$ hidden neurons using Algorithm 2. In the top portion of Figure 7(a), we see a segment

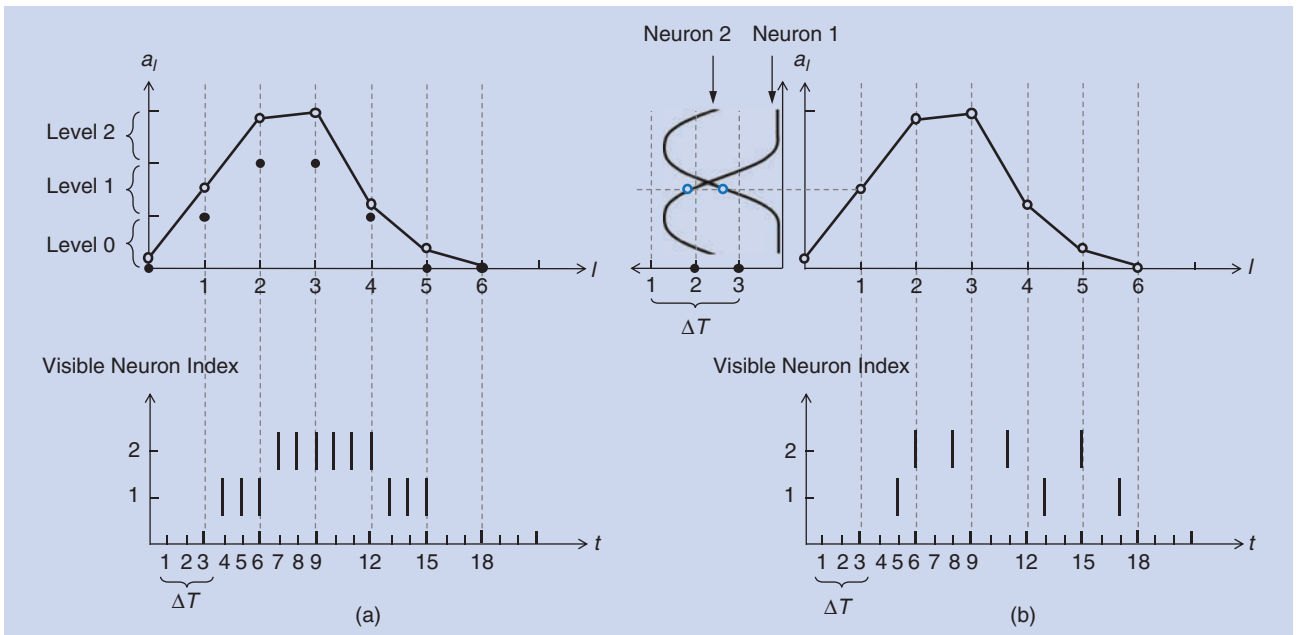


FIGURE 8. Examples of coding schemes with $N_X = 2$ visible neurons and time expansion factor $\Delta T = 3$. (a) With rate coding, each value a_l is discretized into $N_X + 1 = 3$ levels (top), and $\Delta T = 3$ consecutive spikes are assigned to input neuron i for level $i = 1, 2$, and no spikes are assigned otherwise (bottom). (b) With time coding, value a_l is encoded for each visible neuron into zero or one spike, whose timing is given by the value of the corresponding Gaussian receptive field [37].

of the signal and of the prediction for a time window after the observation of the 23,700 plus training samples of the sequence. The corresponding spikes emitted by the SNN [Figure 7(b)] are also shown (top), along with the total number of spikes per time instant [Figure 7(a, bottom)]. The SNN is seen to be able to provide an accurate prediction. Furthermore, the number of spikes, and, hence, the operating energy, depend on the level of activity of the input signal. This demonstrates the potential of SNNs for always-on event-driven applications. As a final note, in this particular example, the hidden neurons are observed to act as a detector of activity versus silence, which facilitates the correct behavior of the visible neurons.

The role of the number N_H of hidden neurons is further investigated in Figure 9, which shows the prediction error as a function of the number of observed training samples for different values of N_H . Increasing the number of hidden neurons is seen to improve the prediction accuracy as long as training is carried out for a sufficiently long time. The prediction error is measured in terms of average mean absolute error (MAE). For reference, we also compare the prediction performance with a persistent baseline (dashed line) that outputs the previous sample, upon quantization to N_X levels for fairness.

Rate versus time encoding

We now discuss the impact of the coding schemes on the online prediction task. We train an SNN with $N_X = 2$ visible neurons and $N_H = 5$ hidden neurons. Figure 10(a) shows the prediction error and Figure 10(b) the number of spikes in a window of 2,500 samples of the input sequence, after the observation of the 17,500 training samples, versus the time expansion factor ΔT . From the figure, rate encoding is seen to be preferable for smaller values of ΔT , while time encoding achieves better prediction error for larger ΔT with fewer spikes and, hence, energy consumption.

This result is a consequence of the different use that the two schemes make of the time expansion ΔT . With rate encoding, a larger ΔT entails a large number of spikes for the neuron encoding the correct quantization level, which provides increased robustness to noise. In contrast, with time encoding, the value ΔT controls the resolution of the mapping between input value a_i and the spiking times of the visible neurons. This demonstrates the efficiency benefits of SNNs that may arise from their unique time encoding capabilities.

Conclusions and open problems

As illustrated by the examples in the previous section, SNNs provide a promising alternative solution to conventional ANNs for the implementation of low-power learning and inference. When using rate encoding, they can approximate the performance of any ANN while also providing a graceful tradeoff between accuracy, on the one hand, and energy consumption and delay, on the other. Most importantly, they have the unique capacity to process time-encoded information, yielding sparse, event-driven, and low-complexity inference and learning solutions.

The recent advances in hardware design reviewed in [5] are motivating renewed efforts to tackle the current lack of well-established direct training algorithms that are able to harness the potential efficiency gains of SNNs. This article has argued that this gap is, at least in part, a consequence of the insistence on the use of deterministic models, which is in turn due to their dominance in the context of ANNs. As discussed, not only can probabilistic models allow the recovery of learning rules that are well known in theoretical neuroscience, but

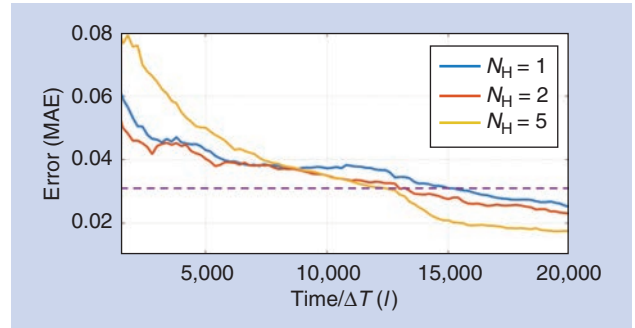


FIGURE 9. Prediction error versus training time for SNNs with $N_X = 9$ visible neurons and $N_H = 1, 2$, and 5 hidden neurons trained via ML learning using Algorithm 2. The dashed line indicates the performance of a baseline persistent predictor that outputs the previous sample (quantized to N_X levels, as described in the text).

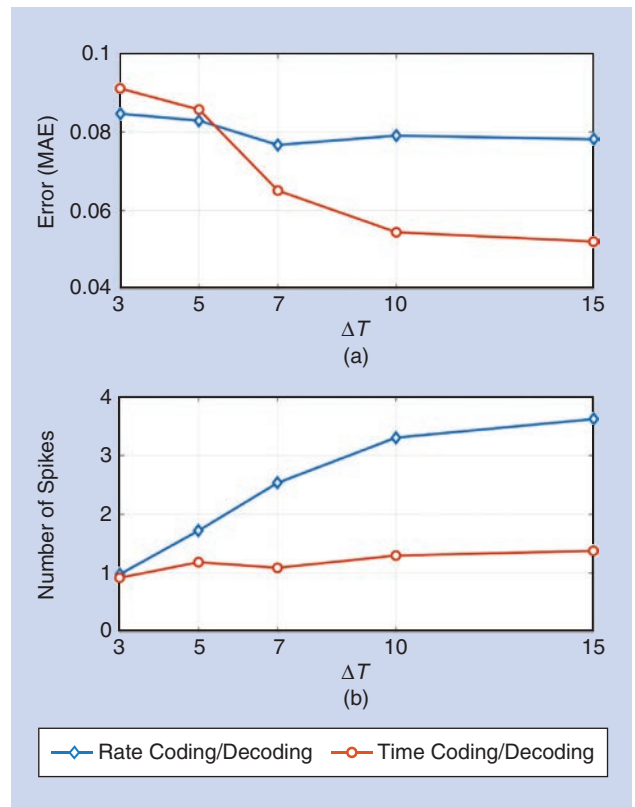


FIGURE 10. An online prediction task based on an SNN consisting of $N_X = 2$ visible neurons and $N_H = 5$ hidden neurons, with rate and time coding schemes: (a) prediction error and (b) number of spikes emitted by the SNN versus the time expansion factor ΔT .

they can also provide a principled framework for the derivation of more general training algorithms. Notably, these algorithms differ significantly from the standard backpropagation approach used for ANNs, owing to their locality coupled with global feedback signaling.

With the main aim of inspiring more research on the topic, this article has presented a review of models and training methods for probabilistic SNNs within a probabilistic signal processing framework. We focused on GLM spiking neuron models, given their flexibility and tractability, and on ML-based training methods. We conclude this article with some discussion on extensions in terms of models and algorithms as well as on open problems.

The SNN models and algorithms we have considered can be extended and modified along various directions. In terms of models, while randomness is defined here at the level of neurons' outputs, alternative models introduce randomness at the level of synapses or thresholds [38], [39]. Furthermore, while the models studied in this article encode information in the temporal behavior of the network within a given interval of time, information can also be retrieved from the asymptotic steady-state spiking rates, which define a joint probability distribution [4], [40], [41]. Specifically, when the GLM (4), (5) has symmetric synaptic weights, i.e., $w_{j,i} = w_{i,j}$, the memory of the synaptic filter is $\tau = 1$, and there is no feedback filter, the conditional probabilities (5) for all neurons define a Gibbs sampling procedure for a Boltzmann machine that can be used for this purpose. As another extension, more general connections among neurons can be defined, including instantaneous firing correlations, and more information, such as a sign, can be encoded in a spike [33]. Finally, while here we focus on signal processing aspects, at a semantic level, SNNs can process logical information by following different principles [11].

In terms of algorithms, the doubly stochastic SGD approach reviewed here for ML training can be extended and improved by leveraging an alternative estimator of the ELBO and its gradients with respect to the variational parameters that is known as the *reparameterization trick* [28]. Furthermore, similar techniques can be developed to tackle other training criteria, such as Bayesian optimal inference [31], reward maximization in reinforcement learning [23], and mutual information maximization for representation learning (see [12] for a discussion in the context of general probabilistic models).

Interesting open problems include the development of metalearning algorithms, whereby the goal is learning how to train or adapt a network to a new task (see, e.g., [41]); the design of distributed learning techniques; and the definition of clear use cases and applications with the quantification of advantages in terms of power efficiency [42]. Another important problem is the design of efficient input/output interfaces between information sources and the SNN, at one end, and between the SNN and actuators or end users, on the other. In the absence of such efficient mechanisms, SNNs risk replacing the so-called

memory wall of standard computing architectures with an input/output wall.

Acknowledgments

This work was supported in part by the European Research Council under the European Union's Horizon 2020 research and innovation program under grant 725731 and by the U.S. National Science Foundation under grant ECCS 1710009. André Grüning (partly) and Brian Gardner (fully) are supported by the European Union's Horizon 2020 Framework Programme for Research and Innovation under the specific grant agreement 785907 (Human Brain Project SGA2).

Authors

Hyeryung Jang (hyeryung.jang@kcl.ac.uk) received her B.S., M.S., and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology, in 2010, 2012, and 2017, respectively. She is currently a research associate in the Department of Informatics, King's College London, United Kingdom. Her recent research interests lie in the mathematical modeling, learning, and inference of probabilistic graphical models, with a specific focus on spiking neural networks and communication systems. Her past research works also include network economics, game theory, and distributed algorithms in communication networks.

Oswaldo Simeone (osvaldo.simeone@kcl.ac.uk) received his M.Sc. degree (with honors) and Ph.D. degree in information engineering from Politecnico di Milano, Italy, in 2001 and 2005, respectively. He is a professor of information engineering with the Centre for Telecommunications Research, Department of Informatics, King's College London, United Kingdom. He is a corecipient of the 2019 IEEE Communication Society Best Tutorial Paper Award, the 2018 IEEE Signal Processing Society Best Paper Award, the 2017 Best Paper by *Journal of Communications and Networks*, the 2015 IEEE Communication Society Best Tutorial Paper Award, and the IEEE International Workshop on Signal Processing Advances in Wireless Communications 2007 and IEEE Wireless Rural and Emergency Communications Conference 2007 Best Paper Awards. He currently serves on the editorial board of *IEEE Signal Processing Magazine* and is a Distinguished Lecturer of the IEEE Information Theory Society. He is a Fellow of the Institution of Engineering and Technology and of the IEEE.

Brian Gardner (b.gardner@surrey.ac.uk) received his M.Phys. degree from the University of Exeter, United Kingdom, in 2011 and his Ph.D. degree in computational neuroscience from the University of Surrey, Guildford, United Kingdom, in 2016. He is a research fellow in the Department of Computer Science, University of Surrey. Currently, his research focuses on the theoretical aspects of learning in spiking neural networks. He is also working as a part of the Human Brain Project and is involved with the implementation of spike-based learning algorithms in neuromorphic systems for embedded applications.

André Grüning (andre.gruening@hochschule-stralsund.de) received his undergraduate degree in theoretical physics from the University of Göttingen, Germany, and his Ph.D. degree in computer science from the University of Leipzig, Germany. He is a professor of mathematics and computational intelligence at the University of Applied Sciences, Stralsund, Germany. He is a visiting member of the European Institute for Theoretical Neuroscience, Paris, France. Previously, he was a senior lecturer (associate professor) in the Department of Computer Science, University of Surrey, Guildford, United Kingdom. He held research posts in computational neuroscience at the Scuola Internazionale Superiore di Studi Avanzati, Trieste, Italy, and in cognitive neuroscience at the University of Warwick, Coventry, United Kingdom. His research concentrates on computational and cognitive neuroscience, especially learning algorithms for spiking neural networks. He is a partner in the Human Brain Project, a European Union Horizon 2020 Flagship Project.

References

- [1] M. Welling, "Intelligence per kilowatt-hour," YouTube, 2018. [Online]. Available: <https://youtu.be/7QhkvG4MUbk>
- [2] H. Paugam-Moisy and S. Bohte, "Computing with spiking neuron networks," in *Handbook of Natural Computing*. Springer-Verlag, 2012, pp. 335–376.
- [3] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Netw.*, vol. 10, no. 9, pp. 1659–1671, 1997.
- [4] M. Davies et al., "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.
- [5] B. Rajendran, A. Sebastian, M. Schmuker, N. Srinivasa, and E. Eleftheriou, "Low-power neuromorphic hardware for signal processing applications. 2019. [Online]. Available: <https://arxiv.org/abs/1901.03690>
- [6] B. Rueckauer and S.-C. Liu, "Conversion of analog to spiking neural networks using sparse temporal coding," in *Proc. IEEE Int. Symp. Circuits and Systems*, Florence, Italy, 2018, pp. 1–5. doi: 10.1109/ISCAS.2018.8351295.
- [7] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Front. Neurosci.*, vol. 10, Nov. 2016. doi: 10.3389/fnins.2016.00508.
- [8] P. O'Connor and M. Welling, "Deep spiking networks. 2016. [Online]. Available: <https://arxiv.org/abs/1602.08323>
- [9] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Front. Neurosci.*, vol. 12, May 2018. doi: 10.3389/fnins.2018.00331.
- [10] P. Dayan and L. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. Cambridge, MA: MIT Press, 2001.
- [11] C. Eliasmith and C. H. Anderson, *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*. Cambridge, MA: MIT Press, 2004.
- [12] O. Simeone, "A brief introduction to machine learning for engineers," *Found. Trends Signal Process.*, vol. 12, no. 3–4, pp. 200–431, 2018.
- [13] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 2018.
- [14] J. W. Pillow, J. Shlens, L. Paninski, A. Sher, A. M. Litke, E. J. Chichilnisky, and E. P. Simoncelli, "Spatio-temporal correlations and visual signalling in a complete neuronal population," *Nature*, vol. 454, no. 7207, Aug. 2008.
- [15] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge, United Kingdom: Cambridge Univ. Press, 2002.
- [16] T. Osogami, "Boltzmann machines for time-series. 2017. [Online]. Available: <https://arxiv.org/abs/1708.06004>
- [17] R. M. Neal, "Connectionist learning of belief networks," *Artif. Intell.*, vol. 56, no. 1, pp. 71–113, 1992.
- [18] F. Gerhard, M. Deger, and W. Truccolo, "On the stability and dynamics of stochastic spiking neuron models: Nonlinear Hawkes process and point process GLMs," *PLoS Comput. Biol.*, vol. 13, no. 2, 2017. doi: 10.1371/journal.pcbi.1005390.
- [19] B. Gardner, I. Sporea, and A. Grüning, "Learning spatiotemporally encoded pattern transformations in structured spiking neural networks," *Neural Comput.*, vol. 27, no. 12, pp. 2548–2586, 2015.
- [20] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks. 2019. [Online]. Available: <https://arxiv.org/abs/1901.09948>
- [21] N. Frémaux and W. Gerstner, "Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules," *Front. Neural Circuits*, vol. 9, Jan. 2016. doi: 10.3389/fncir.2015.00085.
- [22] J. Brea, W. Senn, and J.-P. Pfister, "Matching recall and storage in sequence learning with spiking neural networks," *J. Neurosci.*, vol. 33, no. 23, pp. 9565–9575, 2013.
- [23] B. Rosenfeld, O. Simeone, and B. Rajendran, "Learning first-to-spike policies for neuromorphic control using policy gradients," in *Proc. IEEE Int. Workshop Signal Processing Advances Wireless Communications (SPAWC)*, Cannes, France, 2019. doi: 10.1109/SPAWC.2019.8815546.
- [24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.
- [25] E. L. Bienenstock, L. N. Cooper, and P. W. Munro, "Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex," *J. Neurosci.*, vol. 2, no. 1, pp. 32–48, 1982.
- [26] A. J. Watt and N. S. Desai, "Homeostatic plasticity and STDP: Keeping a neurons cool in a fluctuating world," *Front. Synaptic Neurosci.*, vol. 2, June 2010. doi: 10.3389/fnsyn.2010.00005.
- [27] A. Mnih and K. Gregor, "Neural variational inference and learning in belief networks," in *Proc. Int. Conf. Machine Learning (ICML)*, Beijing, 2014, pp. 1791–1799.
- [28] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes. 2013. [Online]. Available: [arXiv:1312.6114](https://arxiv.org/abs/1312.6114)
- [29] D. J. Rezende and W. Gerstner, "Stochastic variational learning in recurrent spiking networks," *Front. Comput. Neurosci.*, vol. 8, Apr. 2014. doi: 10.3389/fncom.2014.00038.
- [30] G. E. Hinton and A. D. Brown, "Spiking Boltzmann machines," in *Proc. Advances Neural Information Processing Systems (NIPS)*, Denver, CO, 2000, pp. 122–128.
- [31] D. Kappel, S. Habenschuss, R. Legenstein, and W. Maass, "Network plasticity as Bayesian inference," *PLoS Comput. Biol.*, vol. 11, no. 11, 2015. doi: 10.1371/journal.pcbi.1004485.
- [32] F. Leiblein and D. A. Braun, "A reward-maximizing spiking neuron as a bounded rational decision maker," *Neural Comput.*, vol. 27, no. 8, pp. 1686–1720, 2015.
- [33] H. Jang and O. Simeone, "Training dynamic exponential family models with causal and lateral dependencies for generalized neuromorphic computing," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, U.K., 2019, pp. 3382–3386.
- [34] H. Jang, O. Simeone, B. Gardner, and A. Grüning, "An introduction to spiking neural networks: Probabilistic models, learning rules, and applications [supplementary material]," 2019. [Online]. Available: <https://nms.kcl.ac.uk/osvaldo.simeone/spm-supp.pdf>
- [35] J. J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 5, pp. 550–554, 1994.
- [36] H. A. Dau et al., "The UCR time series classification archive," Oct. 2018. [Online]. Available: https://www.cs.ucr.edu/~eamonn/time_series_data_2018
- [37] S. M. Bohte, H. La Poutre, and J. N. Kok, "Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer RBF networks," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 426–435, 2002.
- [38] N. Kasabov, "To spike or not to spike: A probabilistic spiking neuron model," *Neural Netw.*, vol. 23, no. 1, pp. 16–19, 2010.
- [39] H. Mostafa and G. Cauwenberghs, "A learning framework for winner-take-all networks with stochastic synapses," *Neural Comput.*, vol. 30, no. 6, pp. 1542–1572, 2018.
- [40] W. Maass, "Noise as a resource for computation and learning in networks of spiking neurons," *Proc. IEEE*, vol. 102, no. 5, pp. 860–880, 2014.
- [41] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," in *Proc. Advances Neural Information Processing Systems (NIPS)*, Montreal, 2018, pp. 787–797.
- [42] P. Blouw, X. Choo, E. Hunsberger, and C. Eliasmith, "Benchmarking keyword spotting efficiency on neuromorphic hardware. 2018. [Online]. Available: <https://arxiv.org/abs/1812.01739>

Spiking Reservoir Networks

Brain-inspired recurrent algorithms that use random, fixed synaptic strengths



©ISTOCKPHOTO.COM/JUST_SUPER

A class of brain-inspired recurrent algorithms known as *reservoir computing (RC) networks* reduces the computational complexity and cost of training machine-learning models by using random, fixed synaptic strengths. This article offers insights about a spiking reservoir network, the liquid state machine (LSM), the inner workings of the algorithm, the design metrics, and neuromorphic designs. The discussion extends to variations of the LSM that incorporate local plasticity mechanisms and hierarchy to improve performance and memory capacity.

Introduction

Artificial neural networks (ANNs) include a broad class of algorithms that loosely emulate the information processing and learning observed in the central nervous system. Primarily, ANNs learn to perform mapping from an input space (e.g., visual input) to an output space (e.g., object labels). This mapping is achieved through a series of nonlinear transformations parameterized by the summation of a set of weighted synaptic connections. By defining a cost function to measure the error between the network's response and the target response, the synaptic weights can be updated through a process known as *gradient descent*.

While this approach is powerful, the resulting algorithms can be computationally intensive and memory hungry. For example, ResNet-50, a state-of-the-art (SOTA) deep-learning model [1], has 25.6 million parameters, which would require 0.8192 gigabits of memory with 32-bit precision for weight storage alone. Training ResNet-50 on ImageNet would require 14 days with an NVIDIA M40 graphics processing unit (GPU) [2]. However, a growing range of applications that demand real-time, on-device learning on resource-constrained, low-end embedded platforms (e.g., the Internet of Things, edge, sensors) will entail light ANN algorithms.

Brain-inspired algorithms that are computationally light can offer significantly reduced network training time and memory utilization. This article focuses on a subclass of brain-inspired algorithms known as *RC networks*. These networks contain three layers (input, hidden, and readout) of recurrent

NNs (RNNs) that utilize randomly initialized synaptic connections and maintain dynamic information in a high-dimensional hidden layer. Unlike conventional ANNs, in RC networks the synaptic connections from the input layer to the hidden layer and the recurrent connections within the hidden layer are not trained. Only the readout layer of the RC is trained to distinguish input streams based on their representation in the hidden layer. By only training the memory-less readout layer, RC networks have much lower training costs than SOTA networks.

There are two common models of RC networks: the echo state network (ESN) [3] and the LSM proposed by Maass [4], which comprises spiking neurons. The central focus of this article is on spiking reservoir networks, also known as *LSM*. The principles and intuition behind the operation of LSMs generally hold true in other forms of RC networks. As a spiking NN (SNN), the LSM also exhibits several unique properties. SNNs, considered the 3G of NNs are demonstrated to be at least computationally as powerful as networks of threshold and sigmoid neurons [5].

A potential advantage of processing spike trains is that individual spikes can encode richer information based on the temporal dynamics, including an interspike interval and the time to first spike [6], which are not present in the rate-based approximations. Additionally, communicating discrete events through spikes results in robust messaging between neurons with respect to signal noise. The communication overhead is low when it is relayed as binary spike events (an all-or-nothing signal). Moreover, neuromorphic SNNs have demonstrated orders-of-magnitude improvement in energy consumption compared to SOTA nonspiking networks due to their sparse, event-driven communication [7]. Current neuromorphic architectures consume ~20 pJ per spike, which translates to an estimated 100–1,000× smaller energy budgets than the conventional deep NNs realized on GPUs [7]. There have been numerous energy-efficient spiking neuromorphic architecture designs (see [8] for a review), most recently [9] proposed Loihi, which was demonstrated to obtain a three orders of magnitude improvement in energy-delay product compared to conventional CPUs.

To summarize, the advantage of RC networks is their simple architecture, computationally light training without back-propagating error, and short training times. In particular, the LSM as an SNN is capable of using time as a state variable and can be implemented on low-power neuromorphic systems. The availability of such rich features with minimal resource utilization lends these networks to be a natural choice for size-, weight-, and power-constrained platforms.

Inspired by the cerebellum

Though earlier theories categorized learning in the cerebellum, akin to a perceptron, new studies have correlated the three layers in the cerebellum (mossy fibers, granule layers, and Purkinje cells) to the three layers in the LSM [10]. In the cerebellar cortex, classic Marr–Albus theories proposed that Purkinje cells learn associations from high-dimensional representations of information [11]. The authors study how the granule cells support high-dimensional representations of in-

formation conveyed through mossy fibers. In particular, they highlight optimal degrees of sparsity and dimensionality in the granule layer. The inputs to the granule cells are assumed random and not modified during learning; only the Purkinje cells will learn through unsupervised forms of plasticity. The LSM is similar to models of the cerebellar cortex shown in [10] and the high-level concept is represented in Figure 1.

Another region in the brain where information processing has been hypothesized to be similar to the LSM is the prefrontal cortex (PFC) [12]. Based on the idea that a randomly connected network could represent all necessary combinations of task stimuli, referred to as *mixed selectivity*, the authors in [12] studied mixed selectivity using the hidden layer to model the PFC and the readout layer to model the striatum. In [12], the authors discuss the properties of RC networks observed in cortical areas of monkeys, cats, and rats.

LSM

This section provides an overview of a standard LSM model and insights into applying it to real-world problems. The LSM architecture is a three-layer NN consisting of input, hidden, and a readout layers (Figure 1). An intuitive way to understand the operation of an LSM is through an analogy to a pool of water [13]. When objects disturb the surface of the water, they generate ripples that interfere and produce a unique state on the water's surface. An external observer can view the state of the surface and make inferences about the objects, such as where they entered, their size, the timing between sequential objects entering the water, and so on. In this analogy, the objects are the information sent to the hidden layer from the input layer. The pool performs the function of the hidden layer and the external observer has the role of the readout layer, which is discussed in the “Readout Layer” section.

Three neuronal types are used to implement a simple LSM: linear neurons with the identity function, spiking neurons, and logistic neurons (e.g., a sigmoidal activation function). The output of a neuron with the identity function is simply its input and is used in the input layer.

The spiking neuron is used in the hidden layer, although it can also be used for the input and readout layers. One choice for the spiking neuron model is the leaky integrate-and-fire (LIF) neuron [4], whose dynamics are modeled by (assuming a resting potential normalized to 0)

$$\tau_m \frac{\partial V}{\partial t} = -V + I_{\text{ext}} \cdot R, \quad \text{with} \quad V, S = \begin{cases} 0, 1 & \text{if } V \geq V_{\text{th}} \\ V, 0 & \text{otherwise} \end{cases}, \quad (1)$$

where the membrane potential V represents the charge accumulated through the incoming current (I_{ext}) over time. The membrane potential leaks proportional to the time constant τ_m . The output of the LIF neuron is represented by S , which emits a spike ($S = 1$) if the membrane potential reaches the threshold voltage and the membrane potential is reset to its resting state. There are a few factors to consider when choosing the LIF neuron's parameters. The threshold voltage (V_{th}) can control the firing rate of a neuron and the strength of stimulus required to emit a spike. The second parameter is

the membrane time constant (τ_m), which is the product of the membrane capacitance and resistance $\tau_m = R \cdot C$. Changing R will only impact the leakage rate, while changing C will impact both the leakage rate and the integration of current.

The last neuron type we will consider in this review is a neuron with a sigmoidal activation function given by

$$f(x) = \frac{1}{1 + e^{-x}}, \quad (2)$$

which computes a weighted sum on presynaptic inputs denoted as x , and $f(x)$ is the output of the nonlinear transfer function. The sigmoid neuron is used in the readout layer to perform classification. If the goal was to predict a real-valued signal, it would be more appropriate to use a linear neuron.

Input layer

The neurons in the input layer are treated as a set of linear neurons with the identity function (i.e., it forms a direct connection between an input channel and neurons in the hidden layer) [4]. Another option for handling analog signals is to first convert them into temporal spike trains in the input layer [4], [14], which can be achieved by converting the analog values into Poisson spike trains or utilizing other spike-conversion mechanisms. Converting analog values to spikes before the hidden layer adds latency but is essential for the input signal to propagate through the network.

The connections between neurons in the input layer to the hidden layer are random and sparse, where the degree of sparsity depends on the application. In [11], the authors state that

the granule cells need to connect to a sparse number of inputs to produce a unique high-dimensional representation. Using combinatorial math, they show that for N inputs, the degree of synaptic connectivity can sufficiently create an overly complete high-dimensional representation with values as low as 3% of N .

Hidden layer

The hidden layer is a high-dimensional layer of LIF neurons (e.g., higher dimensionality than the input layer). For implementation purposes, the hidden layer will be considered a structured 3D grid of neurons. The hidden layer serves two main roles, to generate a high-dimensional and linearly separable representation of the input space and to capture temporal dynamics of the input space.

In the LSM, hidden layer neurons are considered to have only excitatory (positive) or inhibitory (negative) synapses. The standard ratio of neurons with excitatory connections to neurons with inhibitory connections is 4:1 [4]. The LIF neurons in the hidden layer are recurrently connected to each other. The formation of connections is based on the principle that neurons neighboring each other are more likely to connect [4], so the probability of a connection (i.e., the synaptic strength w_{ij}^{res} between two neurons is greater than 0) is

$$\Pr(w_{ij}^{\text{res}} \neq 0) = C \cdot \exp\left(\frac{-D(i, j)}{\lambda}\right)^2. \quad (3)$$

Two parameters control the formation of recurrent connections in (3): C is a scalar that sets the maximum probability of a

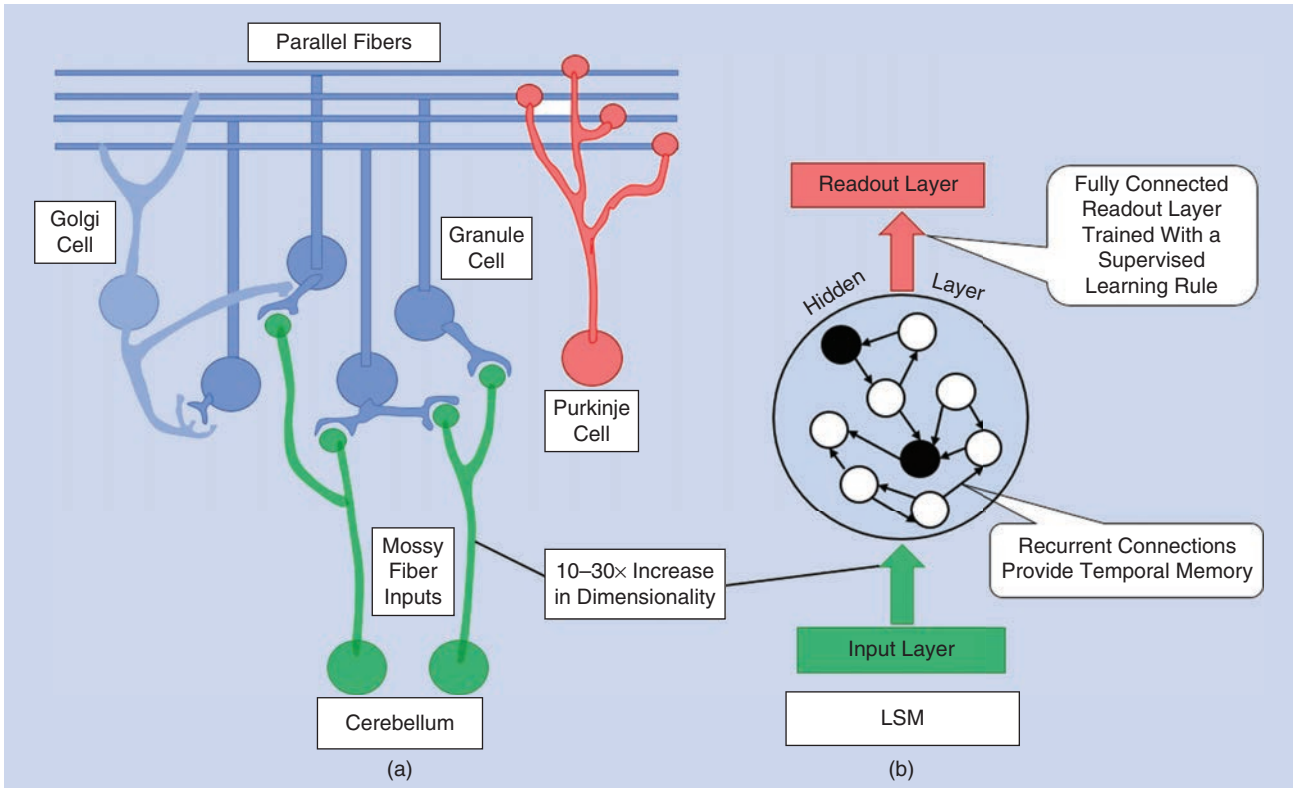


FIGURE 1. The neural circuit formed between mossy fibers, the granule and Golgi cells, and Purkinje cells (a) is similar to the network architecture of the spiking reservoir networks, which is known as the (b) LSM.

connection, and λ controls the probability of a connection being on as the distance between neurons increases. The strength, w_{ij}^{res} , of a formed connection can be drawn from a normal, uniform, or gamma distribution. The mean of the distribution, λ , and C depend on whether the i th (presynaptic neuron) and j th (postsynaptic neuron) are excitatory or inhibitory. In [15], it was shown that maintaining a homogeneous excitability could improve the performance of a standard LSM, which is achieved by making the sum of excitatory synapses and the sum of inhibitory synapses consistent for all neurons through a homeostatic process known as *synaptic scaling* [16]. Computationally, this is equivalent to normalizing the synaptic connections. In the ESN, the recurrent weights are randomly connected with probabilities and strengths drawn from a random distribution (e.g., normal, uniform), which can also be applied to the LSM.

After the input and recurrent connections are initialized, the state of the hidden layer is updated according to (1) and the external current to the neurons is computed as

$$I_h = W_{\text{in}} \cdot u + W_{\text{rec}} \cdot S_h, \quad (4)$$

where the current to the LIF neurons in the hidden layer (I_h) is the weighted sum of the input signal (u) and the spiking activity from other neurons in the hidden layer (S_h).

The synaptic efficacy of the LIF neurons in the hidden layer is regulated through short-term plasticity (STP), a phenomenon where the synaptic efficacy of a neuron's postsynaptic synapses reduces shortly after emitting a spike and recovers slowly. STP acts as a form of hidden memory by reflecting a neuron's recent firing activity, thereby regulating network activity. The STP model used in the original LSM was proposed by [17] and given by

$$R_{n+1} = R_n \cdot (1 - u_{n+1}) \cdot \exp\left(\frac{-\Delta t}{\tau_{\text{rec}}}\right) + 1 - \exp\left(\frac{-\Delta t}{\tau_{\text{rec}}}\right), \quad (5)$$

$$u_{n+1} = u_n \cdot \exp\left(\frac{-\Delta t}{\tau_{\text{facil}}}\right) + U \cdot \left(1 - u_n \cdot \exp\left(\frac{-\Delta t}{\tau_{\text{facil}}}\right)\right), \quad (6)$$

and

$$\text{excitatory postsynaptic potential (EPSP)} = W \cdot R_n \cdot u_n. \quad (7)$$

The magnitude of a spike (EPSP) is a weight between the absolute synaptic efficacy (W , weights in the LSM); utilization of synaptic efficacy, u ; and the fraction of available synaptic efficacy, R . R and u are time-dependent variables, where Δt is the time between the n th and $(n+1)$ th spikes, τ_{rec} controls the recovery from depression, and τ_{facil} controls the rate of decay of u . We provide insights for designing LSM models based on the current literature and a set of guidelines for implementing the hidden layer in RC models, which are applicable to the LSM discussed in [18].

Neuron connectivity

For the neuron parameters, ideally the neurons should match the time scale of the input stream and, in some cases, the target output. For the LIF neuron model described in (1), the steady-state firing rate can be computed as [19]

$$r(I) = \begin{cases} \left[\tau_{\text{ref}} - \tau_m \cdot \log\left(1 - \frac{V_{\text{th}}}{I \cdot R}\right) \right]^{-1}, & \text{if } I \cdot R > V_{\text{th}}, \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

where the firing rate r , corresponding to a steady-state current I , can be computed based on the refractory period τ_{ref} , the membrane time constant τ_m , and the threshold voltage V_{th} . To apply (8) to the LSM neurons, the steady-state current can be approximated by the mean current from the input data. This allows one to predict the firing rate for different sets of parameters, which can be tuned to match a desired firing rate or to ensure that the neurons will be active enough to cause information to propagate through the hidden layer. However, once the hidden layer becomes active, the estimated firing rate rises due to the recurrent connections increasing the average input current. A helpful way to consider the impact of V_{th} and τ_m is by understanding their impact on firing behavior. In particular, a small V_{th} and τ_m emits a spike in response to a few, closely timed presynaptic spikes. As V_{th} becomes larger, the number of presynaptic spikes needed to trigger a postsynaptic spike increases. When τ_m becomes larger, the time window between spikes that will be integrated becomes longer.

Different topologies can be created based on (3). λ can be used to regulate, such as enforcing long excitatory neuron connections with short inhibitory neuron connections or vice versa. The scalar C is used to fine-tune the degree of sparsity in the reservoir. When choosing the parameters to control the formation of recurrent connections, typically one should choose a set of parameters that generate a sparse reservoir. From a practical standpoint, sparse overcomplete representations help facilitate learning and improve robustness to noise in the data [20], and sparsity also tends to result in better performance in terms of accuracy and cost of implementation [18]. In [11] and [20], synaptic degrees of connectivity less than 10% were found to be optimal.

Synaptic strength and plasticity

The strength of the synaptic connections, or sum of synaptic connections, is related to the choice of voltage threshold for the neurons and should be chosen such that the rate of excitability in the hidden layer (firing rate of LIF neurons) is on a desirable time scale. The choice of STP parameters also help to regulate the firing rate of neurons by weakening neurons with high firing rates and can act as a form of hidden memory in the network. Using static synapses without STP has also been shown to have negligible impact on performing small tasks [21].

Hidden layer size

Typically, the size of the hidden layer scales with the complexity of the problem, accounting for factors such as the separability of the data, the dimensionality of the input data, and the length of temporal information the network needs to remember. As stated in [11], an optimal increase in hidden layer dimensionality is approximately 10–50× the size of the input space and results in diminishing returns outside these bounds. Another guideline [18] is that the hidden layer should be large

enough to capture the number of independent signals it has to remember. The authors estimate this by $D^H = D^I \cdot t$, where the dimensionality of the hidden layer (D^H) is equal to the dimensionality of the input space (D^I) times the number of time steps it has to remember, which is the number of time steps on which a prediction is dependent (i.e., the length of a sequence). Larger hidden layer sizes are more likely to form separable representations and minimize the impact of initialization. However, as the size of the hidden layer grows, the LSM can become computationally intensive owing to the memory utilized by the recurrent synaptic connections and is prone to overfitting.

The hyperparameters appropriate for a given task cannot be identified ahead of time as they are based on heuristics. It is therefore important to understand the role of the hyperparameters in the LSM processing. Table 1 provides a summary of the main parameters in the LSM and their impact on the network, which will enable a designer to choose a set of paradigms for testing. From the global test cases, the ideal set of parameters can be selected and fine-tuned for a given task. When optimizing the parameters manually, it often is easier to focus on tuning one parameter at a time. Methods such as automatic grid sweep and evolutionary algorithms can perform efficient searches through the parameter space to identify the best set of parameters for a given application with minimal supervision [22].

Readout layer

The last layer in the LSM is the readout layer. For this review, we will assume it is a layer of neurons (one for each target output variable) with a sigmoidal activation function. However, it

should be noted that the readout layer can also be implemented with linear neurons (typically for regression) or spiking neurons, though learning algorithms for rate-based neurons tend to achieve better performance. When sending the state of the hidden layer to a nonspiking, memoryless readout layer, several states collapse onto each other when using a binary state matrix. This can impact the ability to distinguish different temporal patterns (e.g., if only looking at the last bit between patterns 1, 0, 1 and 1, 1, 1 we would see the same value). There are two methods for averaging the firing activity of the hidden layer: exponential filtering and rate encoding.

Typically, an exponential filtering operation is performed on the output of each neuron in the liquid layer [23]. A technique for computing an exponential filtering of the output is to use a synaptic trace operation at the output of each liquid neuron. This operation is given by

$$\tau_{\text{trace}} \cdot \frac{dX_{\text{trace}}}{dt} = -X_{\text{trace}}, \quad (9)$$

where the synaptic trace (X_{trace}) keeps track of the behavior of the spike activity of a neuron $[S(t)]$. The synaptic trace is incremented by a count of one every time a spike occurs and slowly decays over time, capturing the short-term behavior of each hidden layer neuron.

Another encoding technique is to compute the rate of the spiking neurons activity over a predefined window [21] to provide a measure of a neuron's firing rate. However, using a filtering or trace operation will also provide information about the timing of when a neuron was active, which can be more informative for classification. When performing classification

Table 1. A summary of parameters in designing an effective hidden layer in the LSM and their impact on network behavior.

| Parameter | Description | Comments |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| V_{th} | This controls the rate of response of LIF neurons. | An appropriate firing rate should be based on the time scale of input determined by (8). |
| τ_m | This controls the length of time over which a neuron integrates information and the magnitude of changes in membrane potential. | An appropriate firing rate should be based on the time scale of input (8). Manipulate R and C to adjust the integration of new inputs and the leakage rate separately (1). |
| Input sparsity | This results in mixed selectivity in hidden layer representations and forms an overly complete representation of input data. | In [11], [18], and [20], a small number of connections should be suggested (e.g., less than 10%) to form sufficient representations and reduce the computational complexity. |
| D^H | Larger hidden layers capture more sets of information from the input and have a longer temporal memory for complex problems. | A default is $D^H = 10 \cdot D^I$ [11] or $D^H = t \cdot D^I$ [18]. |
| λ | This controls the likelihood of neurons forming recurrent connections with distant neurons. | Use different λ s to specify the reach of excitatory and inhibitory neurons. |
| C | This controls the sparsity of recurrent connections. | Small degrees of sparsity (e.g., less than 10%) are optimal for performance [20] and computational efficiency [18]. |
| Synaptic strength | This determines the impact of a neuron firing on its neighbor's membrane potential. | The magnitude should be similar to V_{th} to produce the desired firing rates based on (8). |
| U | This is the resting value for a fraction of synaptic efficacy used by an action potential. | A higher U will allow for an action potential to utilize more of the available synaptic efficacy. |
| τ_{facil} | This is the rate at which u decays to its resting value U . | Controlling the available synaptic efficacy, smaller τ_{facil} will have a faster recovery (6). |
| τ_{rec} | This is the rate at which the available synaptic efficacy recovers from depression. | The total available synaptic efficacy, smaller τ_{rec} , will have a faster recovery (5). |

and regression, the neuron activity should be averaged over a timescale matching the frequency of predictions.

The readout layer is fully connected to the hidden layer and trained such that the LSM performs a particular function (e.g., classification, regression) of the inputs by finding a set of output weights that satisfies

$$\mathbf{W}^{\text{out}} = \underset{\mathbf{W}^{\text{out}}}{\operatorname{argmin}} \frac{1}{2} \sum_{\mathbf{X} \in \mathbf{X}_{\text{train}}} (\mathbf{y}(t)(\mathbf{X}) - \hat{\mathbf{y}}(t)(\mathbf{X}))^2, \quad (10)$$

where \mathbf{W}^{out} is the synaptic connections from the hidden layer to the readout layer, \mathbf{X} is the input sample, and $\hat{\mathbf{y}}$ is the network's prediction.

There are two main approaches to solve or iteratively converge to a set of weights that minimizes the cost function. One form of training is referred to as *online training* and occurs when the network is updated iteratively based on the training data [22]. In this approach, the reservoir processes an input sequence, makes a prediction, and then computes the loss and updates the readout weights before moving on to the next sample. Another choice is to solve for the least mean squares solution using the Moore–Penrose pseudo inverse [22], which is achieved by collecting all of the reservoir states associated with training data and solving for the desired output weights. This approach is memory intensive but allows one-shot training of the system.

Metrics for analyzing RC networks

The quality of the LSM can be assessed with two metrics: the separation property [4], which was mathematically formulated in [14], and the Lyapunov exponent [23], given in Table 2. The underlying criteria for the LSM is that the hidden layer representations need to be separable so that a linear classifier can learn the decision boundaries between different classes. There are two implied criteria for good performance, for which the three metrics quantify: inputs belonging to different classes have separable representations (discrimination) and inputs belonging to similar classes result in similar representations in the hidden layer (generalization).

The properties used to assess a reservoir's separation are the intraclass (C_v) variance of reservoir states, which represents patterns from the same class and the interclass variance (C_d), which represents the mean reservoir state of input patterns from different classes. The Lyapunov exponent is calculated based on an input pattern u_j , its nearest neighbor $u_{j'}$, and their respective reservoir states x_j and $x_{j'}$.

In [4], the authors also define an approximation property, stating that the readout layer should be capable of correctly assigning the hidden layer representations to their target classes/values, which is measured by the accuracy of the network. An advantage of assessing the LSM with the Lyapunov exponent is that the user need not have a set of samples grouped into meaningful classes. Although the performance of the network is correlated with these metrics, they do not

directly imply a high accuracy. Better metrics for assessing the quality of the reservoir is an ongoing research topic.

Summary of LSM

In general, the advantage of the LSM, compared to traditional gradient descent-based RNNs, is its computational simplicity regarding training associated with random and fixed synaptic connections. This also discerns the LSM's unique capability to solve multiple tasks without suffering from catastrophic forgetting. When adding a node in the readout layer to perform a new task, the trained weights for that specific node will not conflict or influence the other trained weights. However, the obvious downside to this approach is that randomness is not optimal for a specific task. This has limited the success of RC and the LSM, in particular to complex applications.

Although it has been demonstrated to successfully compete with or beat SOTA [18] on temporal problems when the model was initially proposed, RC did not demonstrate good performance on several challenging benchmarks and has fallen behind SOTA deep NNs such as long short-term memory (LSTM) networks [24]. Few factors limit the applicability of RC approaches to challenging temporal problems that require processing multiple time scales of information or are heavily dependent on spatial information (such as video processing). These limitations are due to the fact that the design and initialization of the hidden layer can have a large impact on performance, the network is a shallow architecture, and that utilizing a simple linear classifier at the readout layer requires the hidden layer to form a linearly separable representation. Because the RC framework is a shallow architecture, it cannot capture long time scales of information, motivating the research community in developing robust and computationally powerful LSM architectures.

Research directions for the LSM

Several research groups have identified that the LSM's bottleneck in performance stems from its limited temporal memory and dependence on initialization. To address these shortcomings, three categories of research have been conducted: introducing a bidirectional readout layer, local plasticity mechanisms, and hierarchical networks. Bidirectional readout layers introduce a random feedback signal from the readout layer to the hidden layer, allowing information from the readout layer to influence the network dynamics that increase the reservoir memory capacity and achieve universal

Table 2. The metrics for assessing the key properties of LSM networks.

| Metric | Equation | Description | Comments |
|---------------------|-----------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|---------------------------------------------------------------|
| Separation property | $\text{Sep}_x(t) = \frac{C_d(t)}{C_v(t) + 1}$ | The ratio of interclass variance to intraclass variance | Quantitative measure of separation and generalization |
| Lyapunov exponent | $\lambda = k \sum_{n=1}^N \ln \left(\frac{\ x_j - x_{j'}\ }{\ u_j - u_{j'}\ } \right)$ | The ratio of RC's representation of two similar inputs $\{x\}$ to the distance between inputs $\{u\}$ | Measures how chaotic the reservoir is when inputs are similar |

computational capabilities [18]. A downside to this approach is that the reservoir dynamics become task dependent, which limits the generalizability and can destabilize the network [18]. Self-organizing LSMs use unsupervised learning rules such as spike-timing dependent plasticity (STDP) to improve the computational capability [13]. The main idea is to use the unsupervised local learning rule known as STDP to optimize the reservoir connections, avoiding the LSM's dependence on initialization. The benefit of using a self-organizing reservoir is that the reservoir can learn to better capture dynamic

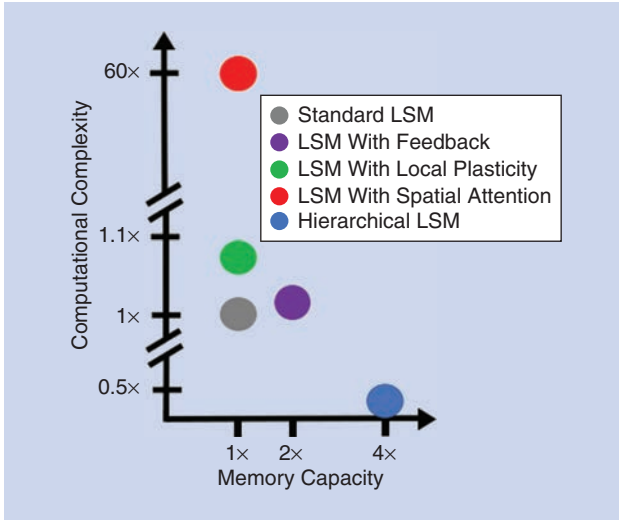


FIGURE 2. The association between techniques to improve the LSM memory capacity and computational complexity. The computational complexity is based on the total number of operations for inference and learning and the number of weights. The reported memory capacity is based on empirical observations of the time steps an LSM model can recall from [25] and [26].

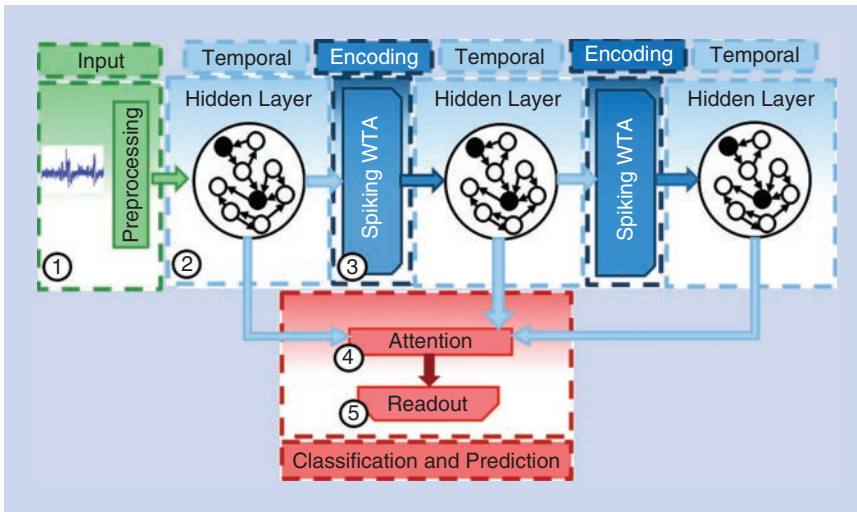


FIGURE 3. The architecture of a deep LSM with three layers. In ①, the input signals are randomly projected to a high-dimensional space in the first hidden layer, and in ②, the hidden layers with random recurrent connections capture temporal information. The spiking WTA layers, which are trained with STDP and encode high-dimensional activity of the hidden layers, are shown in ③, while ④ shows an attention function which condenses the representation of deep-LSM hidden layers for efficient classification. In ⑤, a readout layer is trained to perform a specific task.

information by forming functional groups of related neurons. Hierarchical or deep LSMs use stacked reservoir layers to capture information over multiple time scales [15]. These techniques improve the LSMs' ability to process complex temporal information at the cost of the network's computational complexity (Figure 2). Recently, deep approaches to both LSMs and ESNs have been shown to compete with SOTA performance. As such, the deep-LSM architecture from [15] is shown in Figure 3 and will be explained as a blueprint for implementing hierarchical LSM models.

Deep LSM

A shortcoming of the LSM is that the shallow architecture prevents the network from processing information over longer time scales. In [27], the authors provide evidence that deep networks are computationally more efficient and powerful than a shallow (single-layer) architecture by allowing the network to learn more complex abstractions of the input and to process the input on different time scales in the case of RC networks. To optimize the connectivity and abstraction of information between layers in a deep-LSM model, [15] incorporates encoding layers between subsequent hidden layers.

As can be seen in Figure 3, there are four main layers in the deep-LSM model. The input and hidden layers are the same as the standard LSM implementation. However, there is a bottom-up flow of information between multiple hidden layers (as opposed to a single hidden layer). The high-dimensional representation of information in the hidden layers is projected into a low-dimensional encoding by introducing a spiking winner-takes-all (WTA) layer. Another subsequent hidden layer can then process the low-dimensional projections. The second change from the standard LSM is the attention-modulated readout layer, which is implemented by two separate attention networks.

The inputs to the first layer in the deep-LSM are identical to those of the standard LSM in (4), while the input to the k th unsupervised layer and l th deep hidden layers can be described as

$$I_{E_k}(n) = W_{in}^{E_k} \cdot x_{E_k=k}(n) \quad (11)$$

and

$$I_{L_l}(n) = W_{in}^{L_l} \cdot x_{E_k=l-1}(n) + W_{rec}^{L_l} \cdot x_{L_l}(n-1). \quad (12)$$

As in a standard LSM, the role of the hidden layers is to create high-dimensional, separable representations of the input space and capture dynamic information. However, when choosing the hidden layer size in a deep model, we can use smaller layers than in a standard LSM. For a given task, we will refer to the necessary hidden layer size of a standard LSM as D^H . The dimensionality of hidden layers in the deep-LSM (D^{HD}) is given by

$$D^{HD} = \frac{D^H}{\#Layers}. \quad (13)$$

Because the deep-LSM no longer relies on a single hidden layer to form a separable representation, the size of each hidden layer can be smaller. However, the size of the hidden layers in the deep-LSM should not have a smaller dimensionality than the input space. In between the hidden layers is a WTA layer that forms a low-dimensional encoding of the hidden layer that captures information about the previous layers dynamic behavior, which is fed into a subsequent hidden layer.

The WTA layer is implemented as a layer of feed-forward LIF neurons and is trained with STDP (including synaptic scaling and intrinsic plasticity). To create the WTA behavior where only one neuron can spike for a given time step, global inhibition is used by having a neuron induce a strong inhibitory signal to all other neurons when it spikes, preventing other neurons from also firing and learning the same information.

STDP is a form of Hebbian learning, which increases the synaptic strength of neurons that fire together. If a presynaptic neuron spikes shortly before a postsynaptic neuron, the synaptic strength will be increased. When the timing is reversed and a postsynaptic neuron spikes before a presynaptic neuron, the synaptic strength will be decreased. Mathematically, the change in synaptic strength can be modeled by

$$\Delta W_{ij} = \begin{cases} A^+ \cdot e^{-\frac{\Delta T}{\tau_+}}, & 0 < \Delta T \leq t_{\max} \\ -A^- \cdot e^{-\frac{\Delta T}{\tau_-}}, & 0 \geq \Delta T \geq t_{\min} \end{cases}, \quad (14)$$

where the change in a synaptic strength is based on the timing (ΔT) between the most recent pre- and postsynaptic spikes. A^+ and A^- are scalar learning rates that adjust the magnitude of the synaptic change, while τ_+ and τ_- control the window size of the STDP learning rule.

STDP alone can exhibit runaway dynamics, which result in synaptic strengths saturating. To stabilize the learning dynamics, many works also incorporate homeostatic mechanisms, such as intrinsic plasticity and synaptic scaling [16]. As discussed previously, synaptic scaling maintains the sum of presynaptic connections to remain constant, which is beneficial for synapses trained through STDP because they prevent weights from saturating the bounds of the system and, by itself, can filter out unimportant synaptic connections through the scaling process, as shown in

$$W_{ij} = \frac{W_{ij}}{\sum_{i=1}^N W_{ij}} \cdot \alpha, \quad (15)$$

where α is a scalar to which the summation of all synaptic strengths to a postsynaptic neuron, ($\sum_{i=1}^N W_{ij}$), will be equal.

The second homeostatic mechanism, intrinsic plasticity, regulates a neuron's voltage threshold. Intrinsic plasticity increases a neuron's voltage threshold every time it emits a spike according, which reduces the likelihood that a neuron will be continuously excited and will help to stabilize the firing rate. It will also prevent the neuron potentially inhibiting other neurons

from learning and firing on multiple different input patterns. The threshold is increased according to

$$V_{th} = V_{th0} + \theta, \quad (16)$$

where V_{th0} is the neurons initial threshold voltage and θ is used to regulate the current threshold voltage V_{th} . θ is increased by a fixed increment T every time a postsynaptic neuron spikes and slowly decays back to 0 according to

$$\tau_\theta \cdot \frac{\partial \theta}{\partial t} = -\theta, \quad (17)$$

where τ_θ is a time constant that controls the rate θ decays back to 0.

A starting point for the WTA size is one tenth of the hidden layer size based on the suggestions for the standard LSM producing an increase in dimensionality of at least 10 \times the input layer size. As the size of the WTA layer approaches the size of the hidden layer, the network loses its ability to produce a good high-dimensional representation of the WTA outputs and results in performance drops. For synaptic scaling in the WTA, the sum of connections is typically kept to 1 and, for intrinsic plasticity, the increased voltage threshold after a spike is dependent on the number of neurons in the WTA. The increase should be large enough that it is unlikely for a neuron to fire again in a short time window, during which all of the other neurons have a chance to learn. For the STDP learning rate ($A^{+/-}$), the values should be between 10^{-4} and $10^{-1 \times}$ the average magnitude of the synapses. Generally, the time window of the STDP learning rule should not be so large that the weight update is dependent on input activity from multiple inputs. Suppose there are two classes of inputs streaming one after the other; the network should not learn a combination of both classes.

The last layer is the attention-modulated readout layer. The attention layer is used to focus on the important information in the hidden layer, similar to the process proposed in [28]. There are two attention mechanisms in the deep LSM.

The deep attention network predicts the importance of each layer in the deep LSM according to (18), while the spatial attention network will predict the importance of each neuron location (or group of neurons) according to

$$A_l^{\text{deep}} = \sigma(W_{A^{\text{deep}}} \cdot X_{\text{deep-LSM}}) \quad (18)$$

and

$$A_n^{\text{spatial}} = \sigma(W_{A^{\text{spatial}}} \cdot X_{\text{deep-LSM}}), \quad (19)$$

where $X_{\text{deep-LSM}}$ is the state of the hidden layers in the deep LSM, A_l refers to the attention coefficient for the l th hidden layer, and there are L coefficients, where L is the total number of layers. $W_{A^{\text{deep}}}$ is the learned weights of the deep attention network. Likewise, the coefficients A_n^{spatial} are determined based on the learned weights $W_{A^{\text{two}}}$ and the state of the deep LSM. The spatial attention network predicts N coefficients, where N is the total number of neurons in the hidden layer. Then, based

on the attention coefficients, a weighted sum of the hidden layers is computed as shown in

$$X_F = \left(\sum_{l=1}^L A_l^{\text{deep}} \cdot X_l \right) \odot A^{\text{spatial}}, \quad (20)$$

where the weighted summation of activity in the deep-LSM hidden layers is scaled by A^{spatial} through element-wise multiplication (\odot) and X_F is the final representation of the state of the deep LSM. One downside to this approach is that the attention block does not scale with the size of the deep LSM, which is an area to address in future work.

The final representation X_F is then sent to the output layer, which will compute the final output of the network given by

$$y(n) = \sigma(W_{\text{out}} \cdot X_F), \quad (21)$$

where y_t is the output of the classifier based on the state X_F of the deep LSM at time t . One drawback to the attention-modulated readout layer is that it does not scale well as the depth of the deep LSM grows.

The deep-LSM architecture was demonstrated to outperform a standard LSM and the SOTA models on speech recognition and seizure detection in [15], showing up to an 8% increase in performance over a standard LSM. Recently, the deep LSM was demonstrated to achieve SOTA performance on video activity recognition in comparison to other works using approaches with and without handcrafted features [25]. Similar results have been demonstrated for deep ESNs, where in [29] the authors achieved a SOTA word error rate of 0.0028% on isolated spoken digit recognition. In [27], the authors demonstrate that a deep ESN outperforms SOTA rate-based RNNs such as LSTM networks on polyphonic music tasks in terms of performance and computation time. These results serve as a motivation for deep reservoir-based networks as computationally powerful models for solving temporal problems.

As has been stressed in this article, the advantage of the LSM is its computational efficiency compared to SOTA models. In Table 3, the number of operations, synaptic updates, and memory consumption of the deep LSM, a standard LSM, and an LSTM are compared. The comparison is between the number of operations (estimated by the number of multiplications), weight updates, and memory usage for deep LSM, a standard LSM, and an LSTM with equivalent hidden-layer representations of 1,500 neurons considering a network with 39 inputs and

10 outputs [25]. In the case of LSTM backward pass, we conservatively consider backpropagation over one time step. As can be observed, the deep LSM is computationally more efficient than an LSM with a similar readout, only requiring slightly higher weight updates due to the unsupervised layers. Additionally, both the deep and standard LSM are much less computationally intensive than the LSTM, and less memory intensive, implying an inherent reduction in energy consumption that makes them suitable for embedded platforms.

Neuromorphic LSM implementations

Here we report recent neuromorphic implementations of the LSM and variants that have incorporated on-device learning. For a more in-depth summary of neuromorphic RC architectures, refer to [30]. The digital LSM design in [21] was capable of an online least mean squares learning rule in the readout layer applied to seizure detection. In [13], a field-programmable gate array implementation of the LSM is optimized for energy efficiency with on-device STDP, activity dependent clock gating, and a neuron-gating scheme to minimize power dissipation. The proposed design was demonstrated for speech and image recognition, achieving higher performance and lower energy consumption than a baseline standard LSM and included on-device plasticity for STDP and the readout layer. In [15], a hardware architecture for a memristor-based deep LSM is proposed for speech recognition with on-device learning, which is capable of implementing both STDP and least mean squares in the readout layer. More recently, Intel released a neuromorphic processor, Loihi [9], which is capable of on-device STDP-based learning rules and gated variations for reinforcement learning.

Conclusions and future work

Several research groups have demonstrated the LSM's applicability to real-world problems such as speech recognition [13]–[15], signal alignment [14], image classification [13], and video activity recognition [25]. To address the LSM's limitations on complex tasks due to initialization and short temporal memory capacity, new models with local plasticity rules [13] and hierarchy [25] are being developed. The new models have strong neural plasticity underpinnings and have shown significant performance improvements over the standard LSM. Rigorous analysis of these algorithms is needed to understand how they influence the computational power; extend the temporal memory of RC networks; and optimize methods for hyperparameters, plasticity, and architectures. These studies will help to solidify a set of guidelines for developing models and applying them to appropriate problems.

In particular, a theoretical analysis of the computational complexity and the interplay between different plasticity rules is still missing. New brain-inspired plasticity learning rules that have mathematical models, beyond the STDP, can be exploited to enhance the capabilities of the LSM. Examples of recent techniques in training SNNs that may be of

Table 3. A comparison of the number of operations, weight updates, and memory.

| Network | Synaptic operations (FP) | Synaptic operations (BP) | Weight updates | Memory (Gb) |
|----------------------|--------------------------|--------------------------|----------------|-------------|
| Deep LSM | 110,700 | 15,000 | 17,500 | 0.0035 |
| Deep LSM + attention | 857,200 | 773,506 | 760,500 | 0.0273 |
| LSM | 135,000 | 15,000 | 15,000 | 0.0044 |
| LSM + attention | 2,386,500 | 2,251,500 | 2,251,500 | 0.0764 |
| LSTM | 9,619,500 | 9,675,000 | 9,615,000 | 0.3077 |

FP: forward pass; BP: backward pass.

interest to LSM researchers are three-factor plasticity rules that incorporate neuromodulation and can approximate the learning of gradient descent, reinforcement learning, and novelty detection. Investigating these learning rules can enable the LSM to solve a broader range of problems and realize trainable spiking readout layers.

Reservoir networks are becoming prominent algorithms for resource-constrained platforms due to their low computational complexity and fast training times. As research in both neural plasticity mechanisms and hierarchical reservoir networks progresses, it will open a window for new neuromorphic architectures. Neuromorphic implementations will be crucial for demonstrating the advantages of reservoir networks with on-device learning.

Authors

Nicholas Soures (nms9121@rit.edu) received his B.S. degree in physics from Binghamton University, New York, in 2015 and his M.S. degree in computer engineering from the Rochester Institute of Technology, New York, in 2017, where he is currently pursuing his Ph.D. degree in engineering. His focus is on hardware-driven algorithms and lifelong-learning systems. Other research interests include artificial intelligence, reasoning, and neuromorphic engineering. He is a Student Member of the IEEE.

Dhiresha Kudithipudi (dxkeec@gmail.com) received her B.Tech. degree in electrical and electronics engineering from the Acharya Nagarjuna University, Guntur, India, in 1998, her M.S. degree in computer engineering from Wright State University, Dayton, Ohio, in 2001, and her Ph.D. degree in electrical engineering from the University of Texas at San Antonio, in 2006. She is a professor and director of the Neuromorphic Artificial Intelligence Lab at the Rochester Institute of Technology, New York. She has authored or coauthored several manuscripts and book chapters in the neuromorphic artificial intelligence (AI) field. She serves as an associate editor and guest editor of IEEE and ACM journals. She is passionate about supporting underrepresented and underprivileged students in science, technology, engineering, and mathematics education. Her research interests include AI platforms, brain-inspired neural algorithms, novel computing substrates, and energy-efficient machine intelligence. She is a Senior Member of the IEEE.

References

- [1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2016, pp. 770–778. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf
- [2] Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, and K. Keutzer, "ImageNet training in minutes," in *Proc. ACM 47th Int. Conf. Parallel Processing*, 2018. doi: 10.1145/3225058.3225069. [Online]. Available: <https://dl.acm.org/citation.cfm?id=3225069>
- [3] H. Jaeger, "The 'echo state' approach to analysing and training recurrent neural networks—With an illustrative note," *Fraunhofer Institute for Autonomous Intelligent Systems*, Bonn, Germany, GMD Rep. 148, 2010. [Online]. Available: <https://pdfs.semanticscholar.org/8430/c0b9afa478ae660398704b11dca1221ccf22.pdf>
- [4] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Comput.*, vol. 14, no. 11, pp. 2531–2560, 2002. doi: 10.1162/089976602760407955.
- [5] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Netw.*, vol. 10, no. 9, pp. 1659–1671, 1997. doi: 10.1016/S0893-6080(97)00011-7.

- [6] G. Foffani, M. Morales-Botello, and J. Aguilar, "Spike timing, spike count, and temporal information for the discrimination of tactile stimuli in the rat ventrobasal complex," *J. Neurosci.*, vol. 29, no. 18, pp. 5964–5973, 2009. doi: 10.1523/JNEUROSCI.4416-08.2009.
- [7] E. O. Neftci, C. Augustine, S. Paul, and G. Detorakis, "Event-driven random back-propagation: Enabling neuromorphic deep learning machines," *Front. Neurosci.*, vol. 11, p. 324, 2017. doi: 10.3389/fnins.2017.00324.
- [8] R. A. Nawrocki, R. M. Voyles, and S. E. Shaheen, "A mini review of neuromorphic architectures and implementations," *IEEE Trans. Electron. Devices*, vol. 63, no. 10, pp. 3819–3829, 2016. doi: 10.1109/TED.2016.2598413.
- [9] M. Davies, N. Srinivas, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi et al., "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018. doi: 10.1109/MM.2018.112130359.
- [10] T. Yamazaki and S. Tanaka, "The cerebellum as a liquid state machine," *Neural Netw.*, vol. 20, no. 3, pp. 290–297, 2007. doi: 10.1016/j.neunet.2007.04.004.
- [11] A. Litwin-Kumar, K. D. Harris, R. Axel, H. Sompolinsky, and L. Abbott, "Optimal degrees of synaptic connectivity," *Neuron*, vol. 93, no. 5, pp. 1153–1164, 2017. doi: 10.1016/j.neuron.2017.01.030.
- [12] P. Enel, E. Procyk, R. Quilodran, and P. F. Dominey, "Reservoir computing properties of neural dynamics in prefrontal cortex," *PLoS Comput. Biol.*, vol. 12, no. 6, p. e1004967, 2016. doi: 10.1371/journal.pcbi.1004967.
- [13] Y. Liu, Y. Jin, and P. Li, "Online adaptation and energy minimization for hardware recurrent spiking neural networks," *ACM J. Emerg. Tech. Com.*, vol. 14, no. 1, pp. 1–21, 2018. doi: 10.1145/3145479.
- [14] E. Goodman and D. Ventura, "Spatiotemporal pattern recognition via liquid state machines," in *Proc. IEEE Int. Joint Conf. Neural Networks*, 2006, pp. 3848–3853. [Online]. Available: <https://ieeexplore.ieee.org/document/1716628>
- [15] N. M. Soures, "Deep liquid state machines with neural plasticity and on-device learning," M.S. thesis, Rochester Inst. Technol., Rochester, NY, 2017.
- [16] A. J. Watt and N. S. Desai, "Homeostatic plasticity and STDP: Keeping a neuron's cool in a fluctuating world," *Front. Synaptic Neurosci.*, vol. 2, no. 5, 2010. doi: 10.3389/fnsyn.2010.00005.
- [17] H. Markram, Y. Wang, and M. Tsodyks, "Differential signaling via the same axon of neocortical pyramidal neurons," *Proc. Nat. Acad. Sci.*, vol. 95, no. 9, pp. 5323–5328, 1998. doi: 10.1073/pnas.95.9.5323.
- [18] M. Lukoševičius, "A practical guide to applying echo state networks," in *Neural Networks: Tricks of the Trade*, G. Montavon, G. B. Orr, and K.-R. Müller, Eds. New York: Springer, 2012, pp. 659–686.
- [19] E. Hunsberger and C. Eliasmith, "Spiking deep networks with LIF neurons. 2015. [Online]. Available: <https://arxiv.org/abs/1510.08829>
- [20] O. Barak, M. Rigotti, and S. Fusi, "The sparseness of mixed selectivity neurons controls the generalization–discrimination trade-off," *J. Neurosci.*, vol. 33, no. 9, pp. 3844–3856, 2013. doi: 10.1523/JNEUROSCI.2753-12.2013.
- [21] A. Polepalli, N. Soures, and D. Kudithipudi, "Digital neuromorphic design of a liquid state machine for real-time processing," in *Proc. IEEE Int. Conf. Rebooting Computing*, 2016, 8 pp. doi: 10.1109/ICRC.2016.7738687
- [22] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Comput. Sci. Rev.*, vol. 3, no. 3, pp. 127–149, 2009. doi: 10.1016/j.cosrev.2009.03.005.
- [23] R. Legenstein and W. Maass, "What makes a dynamical system computationally powerful?" *New Directions Stat. Signal Process.*, pp. 127–154, 2007. [Online]. Available: <https://figi-web.tugraz.at/people/maass/psfiles/165.pdf>
- [24] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons. 2018. [Online]. Available: <https://arxiv.org/abs/1803.09574>
- [25] N. Soures and D. Kudithipudi, "Deep liquid state machines with neural plasticity for video activity recognition," *Front. Neurosci.*, vol. 13, pp. 686, 2019. doi: 10.3389/fnins.2019.00686.
- [26] Q. Ma, L. Shen, and G. W. Cottrell, "Deep-ESN: A multiple projection-encoding hierarchical reservoir computing framework. 2017. [Online]. Available: <https://arxiv.org/abs/1711.05255>
- [27] C. Gallicchio, A. Micheli, and L. Pedrelli, "Comparison between deep ESNs and gated RNNs on multivariate time-series prediction. 2018. [Online]. Available: <https://arxiv.org/abs/1812.11527>
- [28] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. Int. Conf. Machine Learning*, 2015, pp. 2048–2057. [Online]. Available: <https://dl.acm.org/citation.cfm?id=3045336>
- [29] C. Gallicchio, A. Micheli, and L. Pedrelli, "Design of deep echo state networks," *Neural Netw.*, vol. 108, pp. 33–47, 2018. doi: 10.1016/j.neunet.2018.08.002.
- [30] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano et al., "Recent advances in physical reservoir computing: A review," *Neural Netw.*, vol. 115, 2019. doi: 10.1016/j.neunet.2019.03.005.

Neuroscience-Inspired Online Unsupervised Learning Algorithms

Artificial neural networks



©ISTOCKPHOTO.COM/JUST_SUPER

Inventors of the original artificial neural networks (ANNs) derived their inspiration from biology [1]. However, today, most ANNs, such as backpropagation-based convolutional deep-learning networks, resemble natural NNs only superficially. Given that, on some tasks, such ANNs achieve human or even superhuman performance, why should one care about such dissimilarity with natural NNs? The algorithms of natural NNs are relevant if one's goal is not just to outperform humans on certain tasks but to develop general-purpose artificial intelligence rivaling that of a human. As contemporary ANNs are far from achieving this goal and natural NNs, by definition, achieve it, natural NNs must contain some “secret sauce” that ANNs lack. This is why we need to understand the algorithms implemented by natural NNs.

Motivated by this argument, we have been developing ANNs that could plausibly model natural NNs on the algorithmic level. In our ANNs, we do not attempt to reproduce many biological details, as in existing biophysical modeling work. Rather, we try to develop algorithms that respect major biological constraints. For example, biologically plausible algorithms must be formulated in the online (or streaming), rather than offline (or batch), setting. This means that input data are streamed to the algorithm sequentially, and the corresponding output must be computed before the next input sample arrives. Furthermore, memory accessible to a biological algorithm is limited so that no significant fraction of previous inputs or outputs can be stored.

Another key constraint is that in biologically plausible NNs, learning rules must be local: a biological synapse can update its weight based on the activity of only the two neurons that the synapse connects. Such locality of the learning rule is violated by most ANNs, including backpropagation-based deep-learning networks. In contrast, our NNs employ exclusively local learning rules. Such rules are also helpful for hardware implementations of ANNs in neuromorphic chips [2], [3].

We derive the algorithms performed by our NNs from optimization objectives. In addition to deriving learning rules for synaptic weights, as is done in existing ANNs, we also derive the architecture, activation functions, and dynamics of neural activity from the same objectives. To do this, we postulate only a cost

function and an optimization algorithm, which, in our case, is alternating stochastic gradient descent/ascent [4]. The steps of this algorithm map to an NN, specifying its architecture, activation functions, dynamics, and learning rules. Viewing both weight and activity updates as the steps of an online optimization algorithm allows us to predict the output of our NNs to a wide range of stimuli without relying on exhaustive numerical simulation.

To derive local learning rules, we employ optimization objectives operating with pairwise similarities of inputs and outputs of an NN rather than individual data points. Typically, our objectives favor similar outputs for similar inputs, hence the name *similarity-matching objectives*. The transformation of dissimilar inputs in the NN depends on the optimization constraints. Despite using pairwise similarities, we still manage to derive online optimization algorithms.

Our focus is on unsupervised learning. This is not a hard constraint but rather a matter of priority. Whereas humans are clearly capable of supervised learning, most of our learning tasks lack big labeled data sets. On the mechanistic level, most neurons lack a clear supervision signal.

Background

Single-neuron online principal component analysis

In a seminal 1982 paper [5], Oja proposed that a biological neuron can be viewed as an implementation of a mathematical algorithm solving a computational objective. Specifically, he modeled a neuron by an online principal component analysis (PCA) algorithm. As PCA is a workhorse of data analysis used for dimensionality reduction, denoising, and latent factor discovery, Oja's model offers an algorithmic-level description of biological NNs.

Oja's single-neuron online PCA algorithm works as follows. At each time step, t , it receives an input data sample, $\mathbf{x}_t \in \mathbb{R}^n$. As our focus is on the online setting, we use the same variable, t , to measure time and index the data points. Then, the algorithm computes and outputs the corresponding top principal component value, $y_t \in \mathbb{R}$:

$$y_t \leftarrow \mathbf{w}_{t-1}^\top \mathbf{x}_t, \quad (1)$$

where $\mathbf{w}_{t-1} \in \mathbb{R}^n$ is the feature vector computed at time step, $t - 1$. Here and ahead, lowercase italic letters are scalar variables, and lowercase boldfaced letters designate vectors. At the same time step t , after computing the principal component, the algorithm updates the (normalized) feature vector with a learning rate η :

$$\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} + \eta(\mathbf{x}_t - \mathbf{w}_{t-1}y_t)y_t. \quad (2)$$

If data are drawn independently from a stationary distribution with a mean vector of zero, the feature vector, \mathbf{w} , converges to the eigenvector corresponding to the largest eigenvalue of input covariance [5].

The steps of the Oja algorithm in (1) and (2) naturally correspond to the operations of a biological neuron. Assuming that the components of the input vector are represented by the activities (firing rates) of the upstream neurons, (1) describes a weighted summation of the inputs by the output neuron. Such weighted summation can be naturally implemented by stor-

ing the components of feature vector, \mathbf{w} , in the corresponding synaptic weights. If the activation function is linear, the output, y_t , is simply the weighted sum. The weight update (2) is a biologically plausible local synaptic learning rule. The first term of the update, $\mathbf{x}_t y_t$, is proportional to the correlation of the presynaptic and postsynaptic neurons' activities. The second term, $\mathbf{w}_t y_t^2$, also local, asymptotically normalizes the synaptic weight vector to one. In neuroscience, synaptic weight updates proportional to the correlation of the presynaptic and postsynaptic neurons' activities are called *Hebbian*.

Minimization of the reconstruction error yields biologically implausible multineuron networks

Next, we would like to build on Oja's insightful identification of biological processes with the steps of the online PCA algorithms by computing multiple principal components using multineuron NNs. Instead of trying to extend the Oja model heuristically, we will derive them by using optimization of a principled objective function. Specifically, we postulate that the algorithm minimizes the reconstruction error, derive an online algorithm optimizing such objective, and map the steps of the algorithm onto biological processes.

In the conventional reconstruction error minimization approach, each data sample, $\mathbf{x}_t \in \mathbb{R}^n$, is approximated as a linear combination of each neuron's feature vector weighted by its activity [4]. Then the minimization of the reconstruction (or coding) error can be expressed as follows:

$$\min_{\mathbf{W}} \sum_{t=1}^T \min_{\mathbf{y}_t} \|\mathbf{x}_t - \mathbf{W}\mathbf{y}_t\|_2^2, \quad (3)$$

where matrix $\mathbf{W} \in \mathbb{R}^{n \times k}$, $k < n$, is a concatenation of feature column vectors and T is both the number of data samples and (in the online setting) the number of time steps.

In the offline setting, a solution to the optimization problem (3) is PCA: the columns of optimum \mathbf{W} are a basis for the k -dimensional principal subspace [6]. Elements of \mathbf{W} could be constrained to avoid unreasonably low or high values. In the online setting, (3) can be solved by alternating minimization [4]. After the arrival of data sample, \mathbf{x}_t , the feature vectors are kept fixed while the objective (3) is minimized with respect to the principal components by running the following gradient-descent dynamics until convergence:

$$\dot{\mathbf{y}}_t = \mathbf{W}_{t-1}^\top \mathbf{x}_t - \mathbf{W}_{t-1}^\top \mathbf{W}_{t-1} \mathbf{y}_t, \quad (4)$$

where $\dot{\cdot}$ is a derivative with respect to a continuous-time variable that runs within a time step, t . Unlike a closed-form output of a single Oja neuron in (1), (4) is iterative.

After the output, \mathbf{y}_t , converges at the same time step, t , the feature vectors are updated according to the following gradient-descent step, with respect to \mathbf{W} on the total objective:

$$\mathbf{W}_t \leftarrow \mathbf{W}_{t-1} + \eta(\mathbf{x}_t - \mathbf{W}_{t-1}\mathbf{y}_t)\mathbf{y}_t^\top. \quad (5)$$

If there were a single output channel, the algorithm in (4) and (5) would reduce to that in (1) and (2), provided that the scalar $\mathbf{W}_{t-1}^\top \mathbf{W}_{t-1}$ is rescaled to unity. In NN implementations of the algorithm in (4) and (5), feature vectors are represented by synaptic

weights and components by the activities of output neurons. Then (4) can be implemented by a single-layer NN, as seen in Figure 1(a), in which activity dynamics converge faster than the time interval between the arrival of successive data samples. The lateral connection weights, $-\mathbf{W}_{t-1}^\top \mathbf{W}_{t-1}$ decorrelate neuronal feature vectors by suppressing activities of correlated neurons.

However, implementing the update in (5) in the single-layer NN architecture, as shown in Figure 1(a), requires nonlocal learning rules, making it biologically implausible. Indeed, the last term in (5) implies that updating the weight of a synapse requires the knowledge of output activities of all other neurons that are not available to the synapse. Furthermore, the matrix of lateral connection weights, $-\mathbf{W}_{t-1}^\top \mathbf{W}_{t-1}$, in the last term of (4) is computed as a Gramian of feedforward weights, clearly a nonlocal operation. This problem is not limited to PCA, and it arises in networks of nonlinear neurons as well [4], [11].

To respect the local learning constraint, many authors constructed biologically plausible single-layer networks using heuristic local learning rules that were not derived from an objective function [12], [13]. However, we think that abandoning the optimization approach creates more problems than it solves. Alternatively, NNs with local learning rules can be derived if one introduces a second layer of neurons [14]. However, such architecture does not map naturally on biological networks.

We outlined how the conventional reconstruction approach fails to generate biologically plausible multineuron networks for online PCA. In the next section, we introduce an alternative approach that overcomes this limitation. Furthermore, this approach suggests a novel view of neural computation, leading to many interesting extensions.

Similarity-based approach to linear dimensionality reduction

In this section, we propose a different objective function for the optimization approach to constructing PCA NNs, which we term *similarity matching*. From this objective function, we

derive an online algorithm implementable by an NN with local learning rules. Then, we introduce other similarity-based algorithms for linear dimensionality reduction that include more biological features, such as different neuron classes.

Similarity-matching objective function

We start by stating an objective function that will be used to derive NNs for linear dimensionality reduction. The similarity of a pair of inputs, \mathbf{x}_t and $\mathbf{x}_{t'}$, both in \mathbb{R}^n , can be defined as their dot product, $\mathbf{x}_t^\top \mathbf{x}_{t'}$. Analogously, the similarity of a pair of outputs, which live in \mathbb{R}^k with $k < n$, is $\mathbf{y}_t^\top \mathbf{y}_{t'}$. *Similarity matching*, as its name suggests, learns a representation where the similarity between each pair of outputs matches that of the corresponding inputs

$$\min_{\mathbf{y}_1, \dots, \mathbf{y}_T} \frac{1}{T^2} \sum_{t=1}^T \sum_{t'=1}^T (\mathbf{x}_t^\top \mathbf{x}_{t'} - \mathbf{y}_t^\top \mathbf{y}_{t'})^2. \quad (6)$$

This offline objective function, previously employed for multidimensional scaling, is optimized by the projections of inputs onto the principal subspace of their covariance (i.e., performing PCA up to an orthogonal rotation). Furthermore, (6) has no local minima other than the principal subspace solution [7], [15]. The similarity-matching objective in (6) may seem like a strange choice for deriving an online algorithm implementable by an NN. In (6), pairs of inputs and outputs from different time steps interact with each other. However, in the online setting, an output must be computed at each time step without accessing inputs or outputs from other time steps. In addition, synaptic weights do not appear explicitly in (6), seemingly precluding mapping onto an NN.

Variable substitution trick

Both of the aforementioned concerns can be resolved by a simple math trick akin to completing the square [16]. We first focus on the cross-term in the expansion of the square in (6), which we call *similarity alignment*. By introducing a new variable, $\mathbf{W} \in \mathbb{R}^{k \times n}$, we can rewrite the cross-term

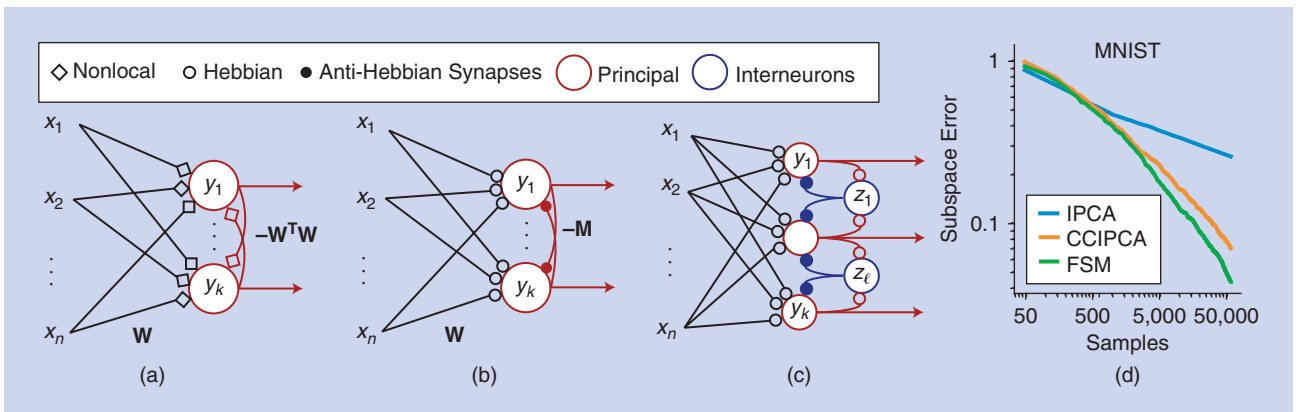


FIGURE 1. (a) The single-layer NN implementation of the multineuron online PCA algorithm derived using the reconstruction approach requires nonlocal learning rules. (b) A Hebbian/anti-Hebbian network derived from similarity matching. (c) A biologically plausible NN with multiple populations of neurons for whitening inputs, derived from a constrained similarity-alignment cost function [7]. (d) The performance of the FSM algorithm, compared to state-of-the-art online PCA algorithms [8], [9] for the MNIST data set reduced to $k = 16$ dimensions (see [10] for details and error definitions). MNIST: Modified National Institute of Standards and Technology database; IPCA: incremental principal component analysis; CCIPCA: candid covariance-free IPCA.

$$-\frac{1}{T^2} \sum_{t=1}^T \sum_{t'=1}^T \mathbf{y}_t^\top \mathbf{y}_{t'} \mathbf{x}_t^\top \mathbf{x}_{t'} = \min_{\mathbf{W} \in \mathbb{R}^{k \times n}} -\frac{2}{T} \sum_{t=1}^T \mathbf{y}_t^\top \mathbf{W} \mathbf{x}_t + \text{Tr} \mathbf{W}^\top \mathbf{W}. \quad (7)$$

To prove this identity, we find optimal \mathbf{W} by taking a derivative of the expression on the right with respect to \mathbf{W} and setting it to zero, and then we substitute the optimal $\mathbf{W}^* = (1/T) \sum_{t=1}^T \mathbf{y}_t \mathbf{x}_t^\top$ back into the expression. Similarly, for the quartic \mathbf{y}_t term in (6)

$$\frac{1}{T^2} \sum_{t=1}^T \sum_{t'=1}^T \mathbf{y}_t^\top \mathbf{y}_{t'} \mathbf{y}_t^\top \mathbf{y}_{t'} = \max_{\mathbf{M} \in \mathbb{R}^{k \times k}} \frac{2}{T} \sum_{t=1}^T \mathbf{y}_t^\top \mathbf{M} \mathbf{y}_t - \text{Tr} \mathbf{M}^\top \mathbf{M}. \quad (8)$$

By substituting (7) and (8) into (6), we get

$$\min_{\mathbf{W} \in \mathbb{R}^{k \times n}} \max_{\mathbf{M} \in \mathbb{R}^{k \times k}} \frac{1}{T} \sum_{t=1}^T [2\text{Tr}(\mathbf{W}^\top \mathbf{W}) - \text{Tr}(\mathbf{M}^\top \mathbf{M}) + \min_{\mathbf{y}_t \in \mathbb{R}^{k \times 1}} l_t(\mathbf{W}, \mathbf{M}, \mathbf{y}_t)], \quad (9)$$

where

$$l_t(\mathbf{W}, \mathbf{M}, \mathbf{y}_t) = -4\mathbf{x}_t^\top \mathbf{W}^\top \mathbf{y}_t + 2\mathbf{y}_t^\top \mathbf{M} \mathbf{y}_t. \quad (10)$$

In the resulting objective function in (9) and (10), optimal outputs at different time steps can be computed independently, making the problem amenable to an online algorithm. The price paid for this simplification is the appearance of the minimax optimization problem in variables \mathbf{W} and \mathbf{M} . Minimization over \mathbf{W} aligns output channels with the greatest variance directions of the input, and maximization over \mathbf{M} diversifies the output by decorrelating output channels similarly to the Gramian, $\mathbf{W}^\top \mathbf{W}$, used previously. This competition in a gradient descent/ascent algorithm results in the principal subspace projection, which is the only stable fixed point of the corresponding dynamics [17].

Online algorithm and NN

We are ready to derive an algorithm for optimizing (6) online. First, we minimize (10) with respect to the output variables, \mathbf{y}_t , while holding \mathbf{W} and \mathbf{M} fixed using gradient-descent dynamics

$$\dot{\mathbf{y}}_t = \mathbf{W} \mathbf{x}_t - \mathbf{M} \mathbf{y}_t. \quad (11)$$

As before, dynamics in (11) converge within a single time step, t , and outputs \mathbf{y}_t . After the convergence of \mathbf{y}_t , we update \mathbf{W} and \mathbf{M} by the gradient descent of (7) and gradient ascent of (8), respectively

$$W_{ij} \leftarrow W_{ij} + \eta(y_i x_j - W_{ij}), \quad M_{ij} \leftarrow M_{ij} + \frac{\eta}{2}(y_i y_j - M_{ij}). \quad (12)$$

The algorithm in (11) and (12), first derived in [17], can be naturally implemented by a biologically plausible NN, as shown in Figure 1(b). Here, activity of the upstream neurons corresponds to input variables. Output variables are computed by the dynamics of activity in (11) in a single layer of neurons. Variables \mathbf{W} and \mathbf{M} are represented by the weights of synapses in feedforward and lateral connections, respectively. The learning rules in (12) are local. That is, the weight update, ΔW_{ij} , for the synapse between the j th input neuron and the i th output neuron depends only on the activities, x_j , of the j th input neuron and y_i of the i th output neuron, and the synaptic weight.

In neuroscience, learning rules in (12) for synaptic weights \mathbf{W} and $-\mathbf{M}$ [here, the minus sign indicates inhibitory synapses; see (11)] are called *Hebbian* and *anti-Hebbian*, respectively.

To summarize this section so far, starting with the similarity-matching objective, we derived a Hebbian/anti-Hebbian NN for dimensionality reduction. The minimax objective can be viewed as a zero-sum game played by the weights of feedforward and lateral connections [16], [18]. This demonstrates that synapses with local updates can still collectively work together to optimize a global objective. A similar, although not identical, NN was proposed by Földiák [12] heuristically. The advantage of our optimization approach is that the offline solution is known.

Although no proof of convergence exists in the online setting, the algorithm in (11) and (12) performs well on large-scale data. A recent paper [10] introduced an efficient, albeit nonbiological, modification of the similarity-matching algorithm, fast similarity matching (FSM), and demonstrated its competitiveness with the state-of-the-art principal subspace projection algorithms in both processing speed and convergence rate, as shown in Figure 1(d). FSM produces the same output \mathbf{y}_t for each input \mathbf{x}_t as similarity matching by optimizing (10) by matrix inversion, $\mathbf{y}_t = \mathbf{M}^{-1} \mathbf{W} \mathbf{x}_t$. It achieves extra computational efficiency by keeping in memory and updating the \mathbf{M}^{-1} matrix rather than \mathbf{M} ; see [10] for suggestions on the implementation of these algorithms. A package with implementations of these algorithms can be found at https://github.com/flatironinstitute/online_psp and https://github.com/flatironinstitute/online_psp_matlab.

Other similarity-based objectives and linear networks

As the algorithm in (11) and (12) and the NN in Figure 1(b) were derived from the similarity-matching objective in (6), they project data onto the principal subspace but do not necessarily recover principal components per se. To derive PCA algorithms, we modified the objective function in (6) to encourage orthogonality of \mathbf{W} [19], [20]. Such algorithms are implemented by NNs of the same architecture, as in Figure 1(b), but with slightly different local learning rules.

Although the similarity-matching NN in Figure 1(b) relies on biologically plausible local learning rules, it lacks biological realism in several other ways. For example, computing output requires recurrent activity that must settle faster than the time scale of the input variation, which is unlikely in biology. To respect this biological constraint, we modified the dimensionality reduction algorithm to avoid recurrence [20].

Another nonbiological feature of the NN in Figure 1(b) is that the output neurons compete with each other by communicating via lateral connections. In biology, such interactions are not direct but are mediated by interneurons. To reflect these observations, we modified the objective function by introducing a whitening constraint

$$\min_{\mathbf{y}_1, \dots, \mathbf{y}_T} -\frac{1}{T^2} \sum_{t=1}^T \sum_{t'=1}^T \mathbf{y}_t^\top \mathbf{y}_{t'} \mathbf{x}_t^\top \mathbf{x}_{t'}, \quad \text{s.t.} \quad \frac{1}{T} \sum_t \mathbf{y}_t \mathbf{y}_t^\top = \mathbf{I}_k, \quad (13)$$

where \mathbf{I}_k is the k -by- k identity matrix. Then, by implementing the whitening constraint using the Lagrange formalism, we derived

NNs where interneurons appear naturally. Their activity is modeled by the Lagrange multipliers, $\mathbf{z}_t^\top \mathbf{z}_{t'}$ [see Figure 1(c)] [7]

$$\min_{\mathbf{y}_1, \dots, \mathbf{y}_T} \max_{\mathbf{z}_1, \dots, \mathbf{z}_T} -\frac{1}{T^2} \sum_{t=1}^T \sum_{t'=1}^T \mathbf{y}_t^\top \mathbf{y}_{t'} \mathbf{x}_t^\top \mathbf{x}_{t'} + \frac{1}{T^2} \sum_{t=1}^T \sum_{t'=1}^T \mathbf{z}_t^\top \mathbf{z}_{t'} (\mathbf{y}_t^\top \mathbf{y}_{t'} - \delta_{t,t'}), \quad (14)$$

where $\delta_{t,t'}$ is the Kronecker delta. Notice how (14) contains the \mathbf{y} - \mathbf{z} similarity-alignment term similar to (7). We can now derive learning rules for the \mathbf{y} - \mathbf{z} connections using the variable substitution trick, leading to the network in Figure 1(c). In addition to dimensionality reduction, such a network can whiten the input data. For details of this and other NN derivations, see [7].

Nonnegative similarity-matching objective and nonnegative independent component analysis

So far, we considered similarity-based NNs comprising linear neurons. But many interesting computations require nonlinearity, and biological neurons are not linear. To construct more realistic and powerful similarity-based NNs, we note that the output of biological neurons is nonnegative (the firing rate cannot be less than zero). Hence, we modified the optimization problem by requiring that the output of the similarity-matching cost function in (6) is nonnegative

$$\min_{\mathbf{y}_1, \dots, \mathbf{y}_T} \frac{1}{T^2} \sum_{t=1}^T \sum_{t'=1}^T (\mathbf{x}_t^\top \mathbf{x}_{t'} - \mathbf{y}_t^\top \mathbf{y}_{t'})^2. \quad (15)$$

Here, the number of output dimensions, k , may be greater than the number of input dimensions, n , leading to a dimensionally expanded representation. Equation (15) can be solved by the

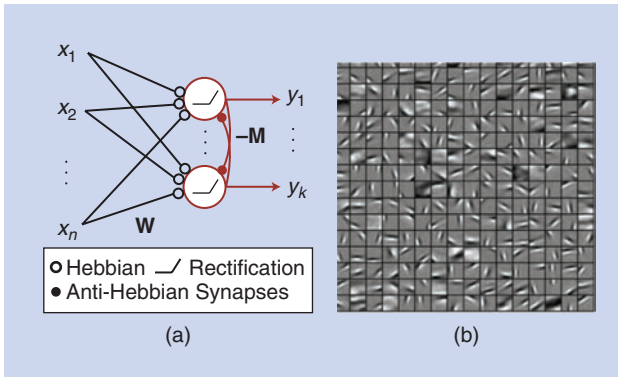


FIGURE 2. (a) A nonlinear Hebbian/anti-Hebbian network derived from NSM. (b) NSM learns edge filters from patches of whitened natural scenes. Learned filters are in small squares. See [21] for details of the simulations.

Table 1. The performance of unsupervised feature learning algorithms.

| Algorithm | Accuracy |
|------------------------|----------|
| Convolutional NSM [24] | 80.42% |
| k-means [22] | 79.60% |
| Convolutional DBN [23] | 78.90% |

We list linear classification accuracy on CIFAR-10 using features extracted by NSM networks. We also list the best single-layer feature extractor (k-means) from an earlier study [22] and a deep-belief network (DBN) [23] on the same task. Detailed comparisons are available in [24].

same online algorithm as in (6) except that the output variables are projected onto the nonnegative domain. Such an algorithm maps onto the same network and learning rules as in (12) [see Figure 1(b)] but with rectifying neurons [21], [25], [26], as shown in Figure 2(a).

A nonnegative similarity-matching (NSM) network learns features that are very different from PCA. For example, when the network is trained on whitened natural scenes, it extracts edge filters [21] (see Figure 2) as opposed to Fourier harmonics expected for a translationally invariant data set. Motivated by this observation, Bahroun and Soltoggio [24] developed a convolutional NSM network with multiple resolutions and used it as an unsupervised feature extractor for subsequent linear classification on the CIFAR-10 data set. They found that NSM NNs are superior to other single-layer unsupervised techniques [24], [27] (see Table 1).

As edge filters emerge also in the independent component analysis (ICA) of natural scenes [28], we investigated a connection of NSM with nonnegative ICA (NICA) used for blind source separation. The NICA problem is to recover independent, nonnegative, and well-grounded (finite probability density function in any positive neighborhood of zero) sources, $\mathbf{s}_t \in \mathbb{R}^d$, from observing only their linear mixture, $\mathbf{x}_t = \mathbf{A}\mathbf{s}_t$, where $\mathbf{A} \in \mathbb{R}^{n \times d}$ and $n \geq d$. Our solution of NICA is based on the observation that NICA can be solved in two steps [29], as shown in Figure 3(a). First, whiten the data and reduce it to d dimensions to obtain an orthogonal rotation of the sources (assuming that the mixing matrix is full rank). Second, find an orthogonal rotation of the whitened sources that yields a nonnegative output, as in Figure 3(a). The first step can be implemented by the whitening network in Figure 1(c). The second step can be implemented by the NSM network, as in Figure 2(a), because an orthogonal rotation does not affect dot-product similarities [25]. Therefore, NICA is solved by stacking the whitening and the NSM networks, as in Figure 3(b). This algorithm performs well compared to other popular NICA algorithms [25], as shown in Figure 3(c).

Nonnegative similarity-based networks for clustering and manifold tiling

NSM can also cluster well-segregated data [21], [30], and for data concentrated on manifolds, it can tile them [31]. To understand this behavior, we analyze the optimal solutions of nonnegative similarity-based objectives. Finding the optimal solution for a constrained similarity-based objective is rather challenging, as has been observed for the nonnegative matrix factorization problem. Here, we introduce a simplified similarity-based objective that allows us to make progress with the analysis and that admits an intuitive interpretation. First, we address the simpler clustering task, which, for highly segregated data, has a straightforward optimal solution. Second, we address manifold learning by viewing it as a soft-clustering problem.

A similarity-based cost function and NN for clustering

The key to our analysis is formulating a similarity-based cost function, an optimization of which will yield an online algorithm and an NN for clustering. The algorithm should assign inputs \mathbf{x}_t

to k clusters based on pairwise similarities and output cluster assignment indices \mathbf{y}_t . To arrive at a cost function, first consider a single pair of data points, \mathbf{x}_1 and \mathbf{x}_2 . If $\mathbf{x}_1^\top \mathbf{x}_2 < \alpha$, where α is a preset threshold, then the points should be assigned to separate clusters—that is, $\mathbf{y}_1 = [1, 0]^\top$ and $\mathbf{y}_2 = [0, 1]^\top$ —setting output similarity, $\mathbf{y}_1^\top \mathbf{y}_2$, to 0. If $\mathbf{x}_1^\top \mathbf{x}_2 > \alpha$, then the points are assigned to the same cluster, such as $\mathbf{y}_1 = \mathbf{y}_2 = [1, 0]^\top$. Such \mathbf{y}_1 and \mathbf{y}_2 are optimal solutions (although not unique) to the following optimization problem:

$$\min_{\mathbf{y}_1 \geq 0, \mathbf{y}_2 \geq 0} (\alpha - \mathbf{x}_1^\top \mathbf{x}_2) \mathbf{y}_1^\top \mathbf{y}_2, \quad \text{s.t. } \|\mathbf{y}_1\|_2 \leq 1, \|\mathbf{y}_2\|_2 \leq 1. \quad (16)$$

To obtain an objective function that would cluster the whole data set of T inputs, we simply sum (16) over all possible input pairs

$$\min_{\mathbf{y}_1 \geq 0, \dots, \mathbf{y}_T \geq 0} \sum_{t=1}^T \sum_{t'=1}^T (\alpha - \mathbf{x}_t^\top \mathbf{x}_{t'}) \mathbf{y}_t^\top \mathbf{y}_{t'} \quad \text{s.t. } \|\mathbf{y}_1\|_2 \leq 1, \dots, \|\mathbf{y}_T\|_2 \leq 1. \quad (17)$$

Does optimization of (17) produce the desired clustering output? This depends on the data set. If a threshold, α , exists such that the similarities of all pairs within the same cluster are greater and similarities of pairs from different clusters are

less than α , then the cost function in (17) is minimized by the desired hard-clustering output, provided that k is greater than or equal to the number of clusters. To solve the objective of (17) in the online setting, we introduce the constraints in the cost via Lagrange multipliers. Using the variable substitution trick, we can derive an NN implementation of this algorithm [31], as shown in Figure 4(a). The algorithm operates with local Hebbian and anti-Hebbian learning rules, whose functional form is equivalent to (12).

Manifold-tiling solutions

In many real-world problems, data points are not well-segregated, but they lie on low-dimensional manifolds. For such data, the optimal solution of the objective in (17) effectively tiles the data manifold [31]. We can understand such optimal solutions using soft clustering (i.e., clustering where each stimulus may be assigned to more than one cluster and assignment indices are real numbers between zero and one). Each output neuron is characterized by the weight vector of incoming synapses, which defines a centroid in the input data space. The response of a neuron is maximum when data fall on the centroid and decay away from it. Manifold-tiling solutions for several data sets are shown in Figure 5.

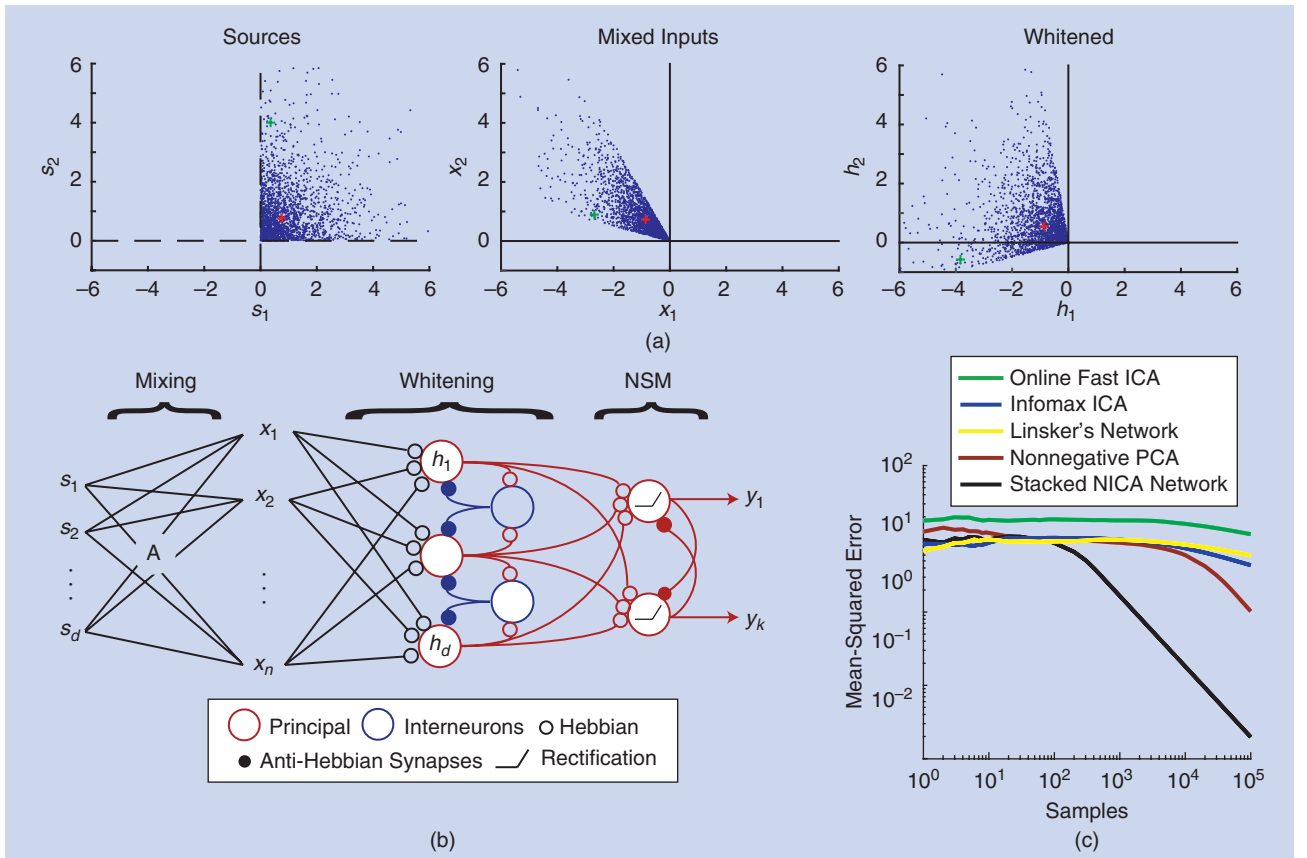


FIGURE 3. (a) An illustration of Plumbley's nonnegative ICA algorithm. Two source channels (left, each blue point shows one source vector, \mathbf{s}_t) are linearly mixed to a 2D mixture, which are the inputs to the algorithm (middle). Whitening (right) yields an orthogonal rotation of the sources. Sources are then recovered by solving the NSM problem. Green and red plus signs track two source vectors across mixing and whitening stages. (b) A stacked network for NICA. The network sees the mixtures \mathbf{x}_t and aims to recover sources \mathbf{s}_t at its output \mathbf{y}_t . (c) The performance of the stacked network for NICA (black) in reconstructing the source vector on a 10-dimensional artificial data set (see [25] for details). The performance metric is the squared error between the network's output and the original sources, averaged over all samples until that point.

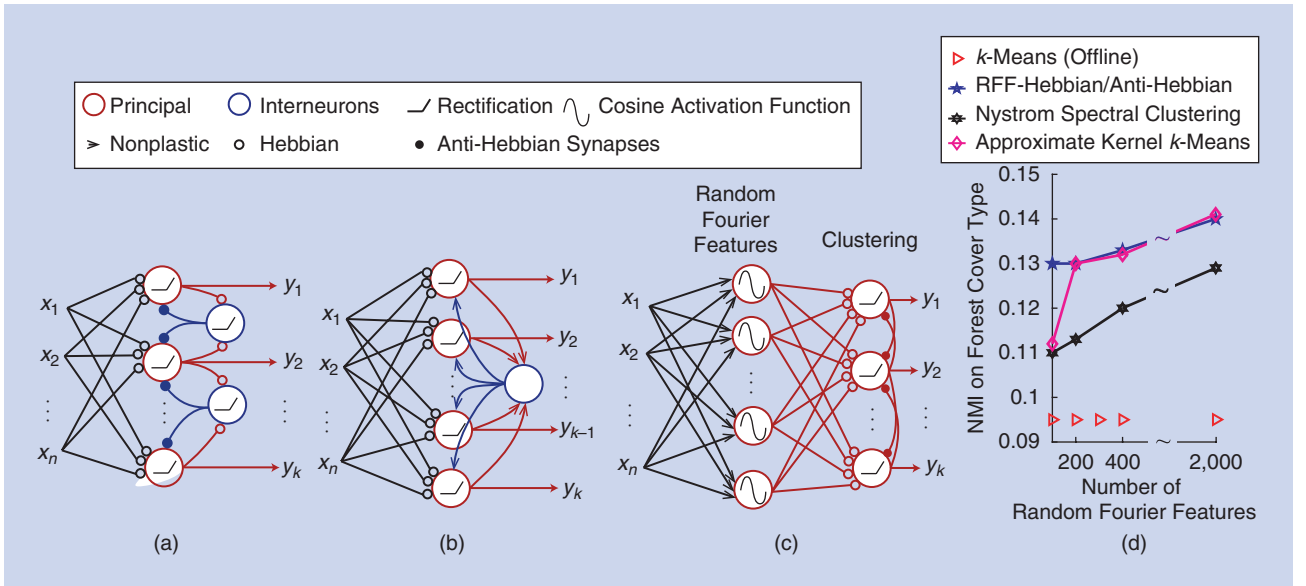


FIGURE 4. Biologically plausible NNs for clustering and manifold learning. (a) A biologically plausible excitatory-inhibitory NN implementation of the algorithm. In this version, anti-Hebbian synapses operate at a faster time scale than Hebbian synapses [31]. (b) The hard and soft k -means networks. Rectified neurons are perfect (hard k -means) or leaky (soft k -means) integrators. They have learned (homeostatic) activation thresholds and ephaptic couplings. (c) When augmented with a hidden nonlinear layer, the presented networks perform clustering in the nonlinear feature space. Shown is the NN of [32], where the hidden layer is formed of random Fourier features (RFFs) [33] to obtain a low-rank approximation to a Gaussian kernel. The two-layer NN operates as an online kernel clustering algorithm. (d) The two-layer NN performs on par to other state-of-the-art kernel clustering algorithms [32]. Shown is performance on forest cover-type data set. The figure is modified from [32]. NMI: normalized mutual information.

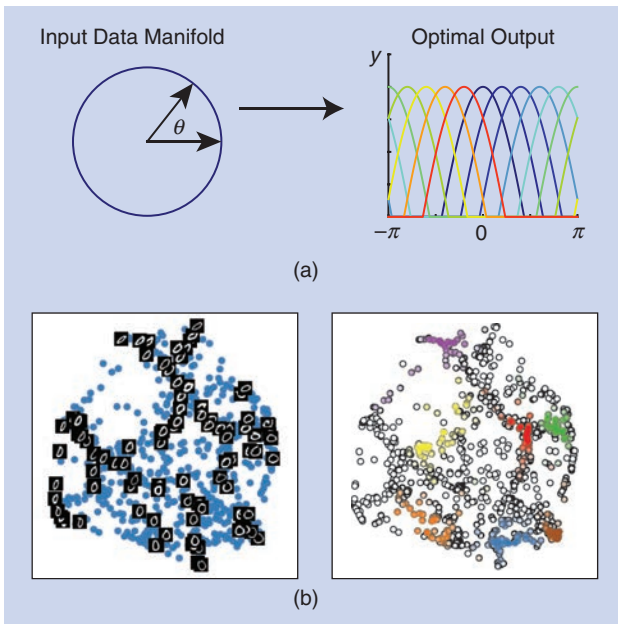


FIGURE 5. Analytical and numerical manifold-tiling solutions of (17) for representative data sets provide accurate and useful representations. (a) A circular manifold (left) is tiled by overlapping localized receptive fields (right). In the continuum limit ($k \rightarrow \infty$), receptive fields are truncated cosines of the polar angle, θ [31]. Similar analytical and numerical results are obtained for a spherical 3D manifold and $SO(3)$ (see [31]). (b) The learning of the manifold of the zero digit from the MNIST data set by tiling the manifold with overlapping localized receptive fields. On the left is 2D linear embedding (PCA) of the outputs. The data are organized according to different visual characteristics of the handwritten digit (e.g., orientation and elongation). On the right are sample receptive fields over the low-dimensional embedding.

We can prove this result analytically by taking advantage of the convex relaxation in the limit of an infinite number of output dimensions (i.e., $k \rightarrow \infty$). Indeed, if we introduce Gramians \mathbf{D} , such that $D_{it} = \mathbf{x}_i^\top \mathbf{x}_t$, and \mathbf{Q} , such that $Q_{it} = \mathbf{y}_i^\top \mathbf{y}_t$, and do not specify the dimensionality of \mathbf{y} by leaving the rank of \mathbf{Q} open, we can rewrite (17) as

$$\min_{\substack{\mathbf{Q} \in \mathcal{CP}^T \\ \text{diag} \mathbf{Q} \leq \mathbf{1}}} -\text{Tr}((\mathbf{D} - \alpha \mathbf{E})\mathbf{Q}), \quad (18)$$

where \mathbf{E} is a matrix whose elements are all ones, and the cone of completely positive $T \times T$ matrices (i.e., matrices $\mathbf{Q} \equiv \mathbf{Y}^\top \mathbf{Y}$ with $\mathbf{Y} \geq 0$) is denoted by \mathcal{CP}^T [34]. Redefining the variables makes the optimization problem convex. For arbitrary data sets, optimization problems in \mathcal{CP}^T are often intractable for large T [34] despite the convexity. However, for symmetric data sets, such as circle, two sphere and $SO(3)$, we can optimize (18) by analyzing the Karush–Kuhn–Tucker conditions [31], as seen in Figure 5(a).

Other similarity-based NNs for clustering and manifold-tiling

A related problem to the objective in (17) is the previously studied convex semidefinite programming relaxation of community detection in graphs [35], which is closely related to clustering. The semidefinite program is related to (18) by requiring the nonnegativity of \mathbf{Q} instead of the nonnegativity of \mathbf{Y}

$$\min_{\substack{\mathbf{Q} \geq 0, \mathbf{Q} \in \mathcal{CP}^T \\ \text{diag} \mathbf{Q} \leq \mathbf{1}}} -\text{Tr}((\mathbf{D} - \alpha \mathbf{E})\mathbf{Q}). \quad (19)$$

Although we chose to present our similarity-based NN approach to clustering and manifold tiling through the cost

function in (17), similar results can be obtained for other versions of similarity-based clustering objective functions. The NSM cost function in (15) and the NN derived from it [Figure 2(a)] can be used for clustering and manifold learning as well [21], [30], [31]. The k -means cost function can be cast into a similarity-based form, and an NN [Figure 4(b)] can be derived for its online implementation [36]. We introduced a soft k -means cost, also a relaxation of another semidefinite program for clustering [37], and an associated NN, as seen in Figure 4(b) [36], and showed that they can perform manifold tiling [38].

The algorithms we discussed operate with the dot product as a measure of similarity in the inputs. By augmenting the presented NNs by an initial random, nonlinear projection layer [Figure 4(c)], it is possible to implement nonlinear similarity measures associated with certain kernels [32]. A clustering algorithm using this idea is shown to perform on par with other online kernel clustering algorithms [32], as seen in Figure 4(d).

Discussion

To overcome the nonlocality of the learning rule in NNs derived from the reconstruction error minimization, we proposed a new class of cost functions called *similarity based*. To summarize, the first term in the similarity-based cost functions,

$$\min_{\mathbf{y}_t, \mathbf{y}_t \in \Omega} \left[-\sum_{t=1}^T \sum_{t'=1}^T \mathbf{y}_t^\top \mathbf{y}_{t'} \mathbf{x}_t^\top \mathbf{x}_{t'} + f(\mathbf{y}_1, \dots, \mathbf{y}_T) \right], \quad (20)$$

is the covariance of the similarity of the outputs and the similarity of the inputs. Hence, the name *similarity-based cost functions*. Previously, such objectives were used in linear kernel alignment [39]. Our key observation is that optimization of objective functions containing such a term in the online setting gives rise to local synaptic learning rules, as in (7) [16].

To derive biologically plausible NNs from (20), one must choose not just the first term but also the function, f , and the optimization constraints, Ω , so that the online optimization algorithm is implementable by biological mechanisms. We and others have identified a whole family of such functions and constraints (see Table 2), some of which were reviewed in this article. As a result, we can relate many features of biological NNs to different terms and constraints in similarity-based cost functions and, hence, give them computational interpretations.

Our framework provides a systematic procedure to design novel NN algorithms by formulating a learning task using similarity-based cost functions. As evidenced by the high-performing algorithms discussed in this article, our procedure of incorporating biological constraints does not impede but rather facilitates the design process by limiting the algorithm search to a useful part of the NN algorithm space. The locality of learning rules in similarity-based NNs makes them naturally suitable for implementation on adaptive neuromorphic systems, which have already been explored in custom analog arrays [3]. For broader use in the rapidly growing world of low-power, spike-based hardware with on-chip learning [2], similarity-based NNs were missing a key ingredient: spiking

Table 2. The current list of objectives, regularizers, and constraints that define a similarity-based optimization problem and are solvable by an NN with local learning.

| Optimization Feature | Biological Feature |
|---------------------------------------|--------------------------------------------------|
| Similarity (anti)alignment | (Anti-)Hebbian plasticity [16], [17] |
| Nonnegativity constraint | Rectifying neuron activation function [18], [21] |
| Sparsity regularizer | Adaptive neural threshold [40] |
| Constrained output correlation matrix | Adaptive lateral weights [7], [18] |
| Constrained PSD output Gramian | Anti-Hebbian interneurons [7] |
| Copositive output Gramian | Anti-Hebbian inhibitory neurons [31] |
| Constrained activity l_1 -norm | Giant interneuron [36] |

neurons. Very recent work [26] developed a spiking version of the NSM and took a step toward neuromorphic applications.

Despite the successes of similarity-based NNs, many interesting challenges remain. First, whereas numerical experiments indicate that our online algorithms perform well, most of them lack global convergence proofs. Even for PCA networks, we can prove linear stability of the desired solution only in the stochastic approximation setting. Second, motivated by biological learning, which is mostly unsupervised, we focused on unsupervised learning. However, supervision, or reinforcement, does take place in the brain. Therefore, it is desirable to extend our framework to supervised, semisupervised, and reinforcement learning settings. Such extensions may be valuable as general purpose machine-learning algorithms.

Third, whereas most sensory stimuli are correlated time series, we assumed that data points at different times are independent. How are temporal correlations analyzed by NNs? Solving this problem is important both for modeling brain function and developing general-purpose machine-learning algorithms. Fourth, another challenge is stacking similarity-based NNs. A heuristic approach to stacking yields promising results [24]. However, except for the NICA problem introduced in the “Nonnegative Similarity-Matching Objective and Nonnegative Independent Component Analysis” section, we do not have a theoretical understanding of how and why to stack similarity-based NNs. Finally, neurons in biological NNs signal each other using all-or-none spikes, or action potentials, as opposed to real-valued signals we considered. Is there an optimization theory accounting for spiking in biological NNs?

Acknowledgments

We thank our collaborators Anirvan Sengupta, Mariano Tepner, Andrea Giovannucci, Alex Genkin, Victor Minden, Sreyas Mohan, and Yanis Bahroun for their contributions; Andrea Soltoggio for discussions; and Siavash Golkar and Alper Erdogan for commenting on the manuscript. This work was supported in part by a gift from the Intel Corporation.

Authors

Cengiz Pehlevan (cpehlevan@seas.harvard.edu) received his undergraduate degrees in physics and electrical engineering from Bogazici University, Istanbul, Turkey, in 2004 and his

doctorate in physics from Brown University, Providence, Rhode Island, in 2011. He was a Swartz Fellow at Harvard University, Cambridge, Massachusetts, a postdoctoral associate at Janelia Research Campus, Ashburn, Virginia, and a research scientist in the neuroscience group at the Flatiron Institute, New York. He is an assistant professor of applied mathematics at the Harvard John A. Paulson School of Engineering and Applied Sciences. His research interests include theoretical neuroscience and neural computation.

Dmitri B. Chklovskii (dchklovskii@flatironinstitute.org) received his Ph.D. degree in theoretical physics from the Massachusetts Institute of Technology, Cambridge. From 1994 to 1997, he was a junior fellow at the Harvard Society of Fellows. He transitioned to neuroscience at the Salk Institute for Biological Studies, San Diego, California. From 1999 to 2007, he was an assistant/associate professor at Cold Spring Harbor Laboratory, New York. Then, as a group leader at Janelia Research Campus, Ashburn, Virginia, he led the team that assembled the largest-at-the-time connectome. He is a group leader for neuroscience at the Flatiron Institute, New York, and a research associate professor at New York University Medical Center. Informed by the function and structure of the brain, his group develops online-learning algorithms for big data.

References

- [1] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, 1958. doi: 10.1037/h0042519.
- [2] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi et al., "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018. doi: 10.1109/MM.2018.112130359.
- [3] J. H. Poikonen and M. Laiho, "A mixed-mode array computing architecture for online dictionary learning," in *Proc. 2017 IEEE Int. Symp. Circuits and Systems*, pp. 1–4.
- [4] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, 1996. doi: 10.1038/381607a0.
- [5] E. Oja, "Simplified neuron model as a principal component analyzer," *J. Math. Biology*, vol. 15, no. 3, pp. 267–273, 1982. doi: 10.1007/BF00275687.
- [6] M. Udeh, C. Horn, R. Zadeh, and S. Boyd, "Generalized low rank models," *Foundations and Trends in Machine Learning*, vol. 9, no. 1, pp. 1–118, 2016. doi: 10.1561/22000000055.
- [7] C. Pehlevan and D. B. Chklovskii, "A normative theory of adaptive dimensionality reduction in neural networks," in *Proc. 28th Int. Conf. Neural Information Processing Systems*, 2015, pp. 2269–2277.
- [8] R. Arora, A. Cotter, K. Livescu, and N. Srebro, "Stochastic optimization for PCA and PLS," in *Proc. 2012 50th Annu. Allerton Conf. Communication, Control, and Computing*, pp. 861–868.
- [9] J. Weng, Y. Zhang, and W.-S. Hwang, "Candid covariance-free incremental principal component analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 8, pp. 1034–1040, 2003.
- [10] A. Giovannucci, V. Minden, C. Pehlevan, and D. B. Chklovskii, "Efficient principal subspace projection of streaming data through fast similarity matching," in *Proc. 2018 IEEE Int. Conf. Big Data (Big Data)*, pp. 1015–1022.
- [11] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999. doi: 10.1038/44565.
- [12] P. Földiák, "Adaptive network for optimal linear feature extraction," in *Proc. Int. 1989 Joint Conf. Neural Networks*, pp. 401–405.
- [13] K. I. Diamantaras and S. Y. Kung, *Principal Component Neural Networks: Theory and Applications*. New York: Wiley, 1996.
- [14] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by V1?" *Vision Res.*, vol. 37, no. 23, pp. 3311–3325, 1997. doi: 10.1016/S0042-6989(97)00169-7.
- [15] R. Ge, J. D. Lee, and T. Ma, "Matrix completion has no spurious local minimum," in *Proc. 30th Int. Conf. Neural Information Processing Systems*, 2016, pp. 2973–2981.
- [16] C. Pehlevan, A. M. Sengupta, and D. B. Chklovskii, "Why do similarity matching objectives lead to Hebbian/anti-Hebbian networks?" *Neural Comput.*, vol. 30, no. 1, pp. 84–124, 2018. doi: 10.1162/neco_a_01018.
- [17] C. Pehlevan, T. Hu, and D. Chklovskii, "A Hebbian/anti-Hebbian neural network for linear subspace learning: A derivation from multidimensional scaling of streaming data," *Neural Comput.*, vol. 27, no. 7, pp. 1461–1495, 2015. doi: 10.1162/NECO_a_00745.
- [18] H. S. Seung and J. Zung, "A correlation game for unsupervised learning yields computational interpretations of Hebbian excitation, anti-Hebbian inhibition, and synapse elimination. 2017. [Online]. Available: <https://arxiv.org/abs/1704.00646>
- [19] C. Pehlevan and D. B. Chklovskii, "Optimization theory of Hebbian/anti-Hebbian networks for PCA and whitening," in *Proc. 2015 53rd Annu. Allerton Conf. Communication, Control, and Computing*, pp. 1458–1465.
- [20] V. Minden, C. Pehlevan, and D. B. Chklovskii, "Biologically plausible online principal component analysis without recurrent dynamics," in *Proc. 2018 52nd Asilomar Conf. Signals, Systems, and Computers*, pp. 104–111.
- [21] C. Pehlevan and D. B. Chklovskii, "A Hebbian/anti-Hebbian network derived from online non-negative matrix factorization can cluster and discover sparse features," in *Proc. 2014 48th Conf. Signals, Systems and Computers*, pp. 769–775.
- [22] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. 14th Int. Conf. Artificial Intelligence and Statistics*, 2011, pp. 215–223.
- [23] A. Krizhevsky and G. Hinton, "Convolutional deep belief networks on CIFAR-10," unpublished.
- [24] Y. Bahroun and A. Soltoggio, "Online representation learning with single and multi-layer Hebbian networks for image classification," in *Proc. Int. Conf. Artificial Neural Networks*, 2017, pp. 354–363.
- [25] C. Pehlevan, S. Mohan, and D. B. Chklovskii, "Blind nonnegative source separation using biological neural networks," *Neural Comput.*, vol. 29, no. 11, pp. 2925–2954, 2017. doi: 10.1162/neco_a_01007.
- [26] C. Pehlevan, "A spiking neural network with local learning rules derived from nonnegative similarity matching," in *Proc. 2019 IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pp. 7958–7962.
- [27] Y. Bahroun, E. Hunsicker, and A. Soltoggio, "Building efficient deep Hebbian networks for image classification tasks," in *Proc. Int. Conf. Artificial Neural Networks*, 2017, pp. 364–372.
- [28] A. J. Bell and T. J. Sejnowski, "The 'independent components' of natural scenes are edge filters," *Vision Res.*, vol. 37, no. 23, pp. 3327–3338, 1997. doi: 10.1016/S0042-6989(97)00121-1.
- [29] M. Plumbley, "Conditions for nonnegative independent component analysis," *IEEE Signal Process. Lett.*, vol. 9, no. 6, pp. 177–180, 2002. doi: 10.1109/LSP.2002.800502.
- [30] D. Kuang, C. Ding, and H. Park, "Symmetric nonnegative matrix factorization for graph clustering," in *Proc. 2012 SIAM Int. Conf. Data Mining*, pp. 106–117.
- [31] A. Sengupta, C. Pehlevan, M. Tepper, A. Genkin, and D. Chklovskii, "Manifold-tiling localized receptive fields are optimal in similarity-preserving neural networks," in *Proc. 2018 IEEE Int. Conf. Big Data (Big Data), Advances in Neural Information Processing Systems*, pp. 7080–7090.
- [32] Y. Bahroun, E. Hunsicker, and A. Soltoggio, "Neural networks for efficient nonlinear online clustering," in *Proc. Int. Conf. Neural Information Processing*, 2017, pp. 316–324.
- [33] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. 20th Int. Conf. Neural Information Processing Systems*, 2008, pp. 1177–1184.
- [34] A. Berman and N. Shaked-Monderer, *Completely Positive Matrices*. Singapore: World Scientific, 2003.
- [35] T. T. Cai and X. Li, "Robust and computationally feasible community detection in the presence of arbitrary outlier nodes," *Ann. Statist.*, vol. 43, no. 3, pp. 1027–1059, 2015. doi: 10.1214/14-AOS1290.
- [36] C. Pehlevan, A. Genkin, and D. B. Chklovskii, "A clustering neural network model of insect olfaction," in *Proc. 2017 51st Asilomar Conf. Signals, Systems, and Computers*, pp. 593–600.
- [37] B. Kulis, S. Basu, I. Dhillon, and R. Mooney, "Semi-supervised graph clustering: A kernel approach," *Mach. Learn.*, vol. 74, no. 1, pp. 1–22, 2009. doi: 10.1007/s10994-008-5084-4.
- [38] M. Tepper, A. M. Sengupta, and D. Chklovskii, "Clustering is semidefinitely not that hard: Nonnegative SDP for manifold disentangling," *J. Mach. Learning Res.*, vol. 19, no. 1, pp. 3208–3237, 2018.
- [39] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. S. Kandola, "On kernel-target alignment," in *Proc. 14th Int. Conf. Neural Information Processing Systems: Natural and Synthetic*, 2002, pp. 367–373.
- [40] T. Hu, C. Pehlevan, and D. B. Chklovskii, "A Hebbian/anti-Hebbian network for online sparse dictionary learning derived from symmetric matrix factorization," in *Proc. 2014 48th Asilomar Conf. Signals, Systems and Computers*, pp. 613–619.

Bipin Rajendran, Abu Sebastian, Michael Schmuker,
Narayan Srinivasa, and Evangelos Eleftheriou

Low-Power Neuromorphic Hardware for Signal Processing Applications

A review of architectural and system-level design approaches



©ISTOCKPHOTO.COM/JUST_SUPER

Machine learning has emerged as the dominant tool for implementing complex cognitive tasks that require supervised, unsupervised, and reinforcement learning. While the resulting machines have demonstrated in some cases even superhuman performance, their energy consumption has often proved to be prohibitive in the absence of costly supercomputers. Most state-of-the-art machine-learning solutions are based on memoryless models of neurons. This is unlike the neurons in the human brain that encode and process information using temporal information in spike events. The different computing principles underlying biological neurons and how they combine together to efficiently process information is believed to be a key factor behind their superior efficiency compared to current machine-learning systems.

Inspired by the time-encoding mechanism used by the brain, third-generation spiking neural networks (SNNs) are being studied for building a new class of information processing engines. Modern computing systems based on the von Neumann architecture, however, are ill suited for efficiently implementing SNNs since their performance is limited by the need to constantly shuttle data between physically separated logic and memory units. Hence, novel computational architectures that address the von Neumann bottleneck are necessary to build systems that can implement SNNs with low energy budgets. In this article, we review some of the architectural and system-level design aspects involved in developing a new class of brain-inspired information processing engines that mimic the time-based information encoding and processing aspects of the brain.

Introduction

While machine-learning algorithms based on deep NNs have demonstrated humanlike or even superhuman performance in several complex cognitive tasks, a significant gap exists between the energy and efficiency of the computational systems that implement these algorithms compared to the human brain. Most of these algorithms run on conventional computing systems, such as central processing units (CPUs),

graphical processing units (GPUs), and field programmable gate arrays. Recently, digital or mixed-signal application-specific integrated circuits (ASICs) have also been developed for machine-learning acceleration. However, as Moore's law scaling is coming to an imminent end, the performance and power efficiency gains from the technology scaling of these conventional approaches are diminishing. Thus, there are significant research efforts worldwide toward developing a profoundly different approach to computing for artificial intelligence (AI) applications inspired by biological principles.

In the traditional von Neumann architecture, a powerful processing unit operates sequentially on data fetched from memory. In such machines, the von Neumann bottleneck is defined as the limitation on performance arising from the chokepoint between computation and data storage. Hence, the research focus has been not only on designing new AI algorithms, device technologies, integration schemes, and architectures but on overcoming the CPU/memory bottleneck in conventional computers. SNNs are the third generation of artificial neuron models that leverage the key time-based information encoding and processing aspects of the brain. Neuromorphic computing platforms aim to efficiently emulate SNNs in hardware by distributing both computation and memory among a large number of relatively simple computation units, namely, the neurons, each passing information via asynchronous spikes to hundreds or thousands of other neurons through synapses [1].

The event-driven characteristics of SNNs have led to highly efficient computing architectures with collocated memory and processing units, significantly increased parallelism, and reduced energy budgets. Such architectures have been demonstrated in neuromorphic implementations such as SpiNNaker, from the University of Manchester [2]; IBM's TrueNorth [3]; Intel's Loihi [4]; BrainScaleS, built by Heidelberg University [5]; NeuroGrid, from Stanford University [6]; INI Zürich's DYNAP [7]; and ODIN, created by Catholic University Louvain [8]. Moreover, breakthroughs in the area of nanoscale memristive devices have enabled further improvements in the area and energy efficiency of mixed digital-analog implementations of synapses and spiking neurons [9], [10]. In this article, we provide a high-level description of the design objectives and approaches that are currently being pursued for building energy-efficient neuromorphic computing platforms.

Information processing in the brain

Computing in the brain follows a completely different paradigm than today's conventional computing systems. Whereas conventional systems are optimized to transmit and modify numerical representations of data, the brain operates on timed events called *action potentials* or *spikes*. Neurons receive these spikes via synapses, which convert them into small changes in the cell's membrane potential. The neuron integrates these changes in potential over time, and, under certain conditions—for instance, when many spikes arrive within a short time—the

neuron emits a spike. Spikes can be considered messages in the computing sense, except that they carry no information other than their time of generation and their source. Computing in the brain can thus be described as fully event driven, nonblocking, and imperatively concurrent, with very lightweight compute nodes (the neurons), that communicate via tiny messages, the spikes.

There are about 10^{11} neurons in the human brain, and it is estimated that there are about 5,000–10,000 synapses per neuron in the human neocortex. Thus, connectivity is sparse, with a neuron receiving input from $10^{-6}\%$ of all other neurons (probably even less, considering that an axon may form multiple synapses on the same dendrite). At the same time, the total number of connections is huge, in the order of 10^{15} . This is vastly different from the low fan-out connectivity that is common in conventional computers.

Signal encoding in the brain

The event-driven nature of computing also applies to stimulus encoding by the sensory organs. Generally, sensory coding emphasizes changes in the stimulus rather than accurate coding of constant levels. These changes can be considered events that are translated into spikes for downstream processing. For example, ganglion cells in the retina transmit a spike if the change in local contrast in their receptive field exceeds a threshold. They then adapt to the new level of local contrast. More spikes are produced

only when the local contrast rises further. This encoding scheme has three advantages over uniformly spaced signal sampling in conventional signal processing.

- 1) It produces a sparse code that transmits information only when the input signal is changing.
- 2) It is not limited by a fixed sample rate in the maximum frequency it can encode.
- 3) It can lead to extremely reactive systems since the latency for feature extraction is not limited by the time between two samples but only the minimum delay between two events, which is usually much shorter.

In neuromorphic devices, this encoding approach has been implemented via a set of thresholds that trigger events to be fired upon the signal crossing them [Figure 1(b)] [11].

Learning in the brain

The connections in the brain are not fixed but can change as a function of their activity; this process is often called *learning*. Fundamental principles of synaptic changes have been uncovered that depend on the timing of spikes fired in the pre- and postsynaptic cells, thus termed *spike-timing-dependent plasticity* (STDP) [12]. STDP implements a form of Hebbian learning, where a synapse is strengthened (*potentiated* in neuroscience terminology) if the presynaptic spike arrives within a certain time window preceding the spike of the postsynaptic cell or is weakened (depressed) if the temporal order is reversed [Figure 1(c)]. Pre- and postsynaptic

Computing in the brain follows a completely different paradigm than today's conventional computing systems.

timing are thus two factors that determine the change of a synaptic weight.

In biology, the amount and direction of weight change are often influenced by neuromodulators, such as dopamine or noradrenaline, which are released as a function of the reward received by the organism. Neuromodulators are thus a third factor in models of synaptic learning. They allow for the construction of powerful spiking learning rules that, for example, implement reinforcement learning [13]. Taken together, the brain's massively parallel, event-driven computing and its extraordinary connectivity are probably the basis for its extremely high efficiency in signal processing, inference, and control. Furthermore, the action potentials used for communication as well as the synaptic currents in the brain have much smaller magnitudes than those of the electrical signals in silicon computers, as signals in the brain are encoded by the flow of a relatively smaller number of charge carriers (such as sodium, potassium, and calcium ions) through ion channels with highly selective, stochastic, and nonlinear conductance characteristics. As a result of all of these contributing factors, the brain is estimated to consume about 20 W, even during demanding tasks like mastering a multiplayer online computer game, whereas a conventional platform has been reported to use about 128,000 CPUs and 256 GPUs to achieve competitive performance (see <https://blog.openai.com/openai-five>).

This fundamental difference in computing architecture implies that porting algorithms from conventional machines to spike-based approaches can have only limited success. New algorithms are required that embrace the fundamentally event-based nature of sensing, learning, and inference in the brain to

leverage the full potential of spiking networks accelerated by neuromorphic hardware.

Building blocks of neuromorphic systems

Although biological neurons and synapses exhibit a wide variety of complex dynamical behaviors, most hardware designs need to mimic only the key aspects that are essential for computation. At a high level, this includes the integrate-and-fire (I&F) dynamics of neurons and the spike-triggered communication and plasticity of synapses.

The central aspects of the I&F dynamics of neurons are described by the Hodgkin–Huxley equations, which incorporate the voltage-dependent dynamics of sodium, potassium, and leaky ion channels to determine the evolution of the membrane potential. While biological neurons exhibit more complex behaviors, such as burst firing, chirping, postinhibitory rebound, and spike-frequency adaptation, and although the computational significance of these has not yet been clearly established, there have been hardware designs that mimic some of these behaviors, using both digital CMOS and subthreshold analog CMOS circuits.

Model-order reduction techniques have been used to reduce the complexity of the dynamical equations that describe these behaviors; some of the notable examples being the second-order Izhikevich model, the adaptive exponential I&F model, and the linear leaky I&F (LIF) model. These are simpler to implement in hardware and are more commonly used in large-scale neuromorphic designs. We describe the LIF model here, as it is the most commonly used spiking neuron model. The membrane potential $V(t)$ evolves according to the differential equation:

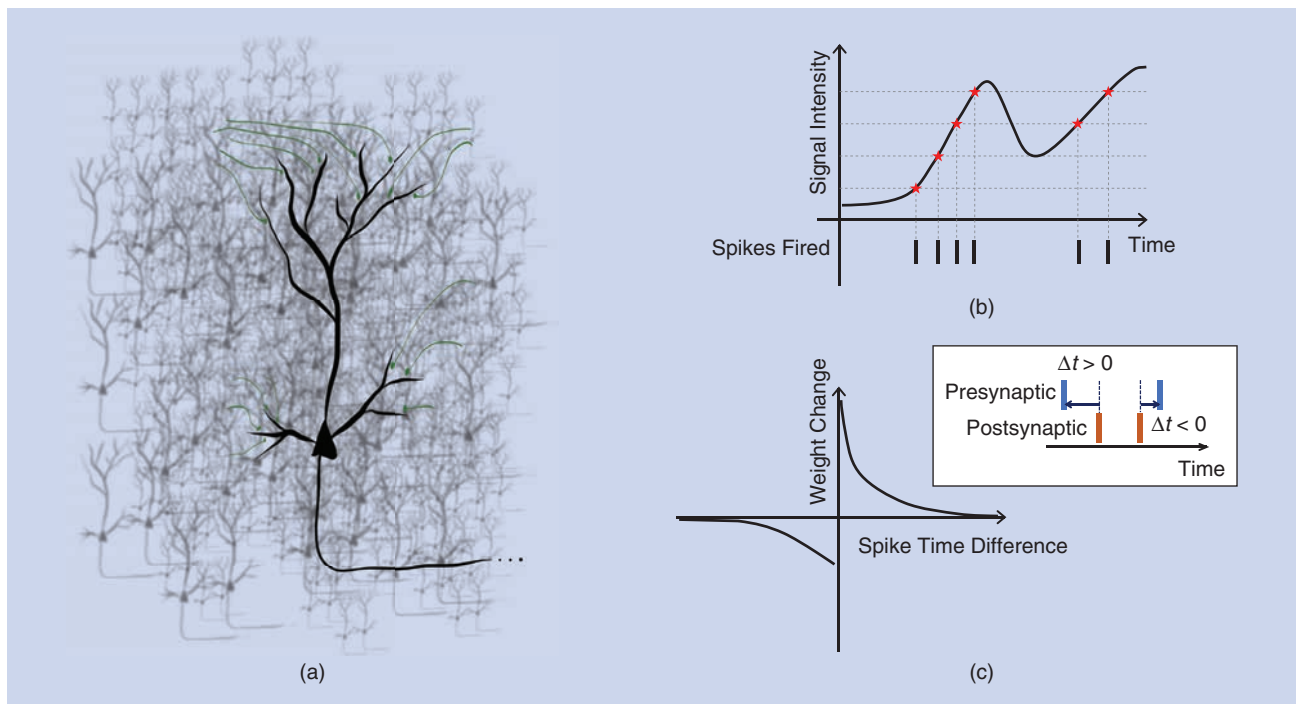


FIGURE 1. (a) The brain employs a massively parallel computational network comprising $\sim 10^{11}$ neurons, each with some 5,000–10,000 synapses, in a system of distributed, asynchronous, event-driven computing that uses (b) event-driven signal encoding and (c) event-driven weight updates.

$$C \frac{dV(t)}{dt} = -g_L(V(t) - E_L) + I_{\text{syn}}(t),$$

$$V(t) \leftarrow E_L, \text{ if } V(t) > V_T, \quad (1)$$

where the input synaptic current $I_{\text{syn}}(t)$ is integrated across a leaky capacitor until the voltage exceeds a threshold V_T , when a spike is issued, and the membrane potential is reset to its resting value E_L . C and g_L model the membrane's capacitance and leak conductance, respectively. The refractory period seen in biological neurons can be implemented by holding the membrane potential at $V(t) = E_L$, preventing current integration during that period.

Synaptic communication is triggered by the arrival of a spike, causing a current to flow into downstream neurons. This is also a highly complex process in the brain, involving the release of neurotransmitter molecules at the axon terminal, which then diffuse across the synaptic cleft and bind with receptor molecules at the dendrite, causing ionic currents to flow into the downstream neurons. These aspects are not modeled in most hardware implementations. Instead, the current through a synapse with strength w is calculated as

$$I_{\text{syn}}(t) = w \times \sum_i \alpha(t - t^i), \quad (2)$$

where t^i represents the time of arrival of spikes at the axon terminal and $\alpha(t)$ is a synaptic current kernel; $\alpha(t) = (e^{-t/\tau_1} - e^{-t/\tau_2})$, $\alpha(t) = (t/\tau)e^{-t/\tau}$, and $\alpha(t) = \tau\delta(t)$ are some

commonly used synaptic kernels. Note that this form of synaptic transmission also assumes that the current is independent of the postsynaptic neuron's potential, unlike in biology. Finally, to implement synaptic plasticity, the weight w is updated based on learning rules implemented in peripheral circuits, again in a spike-triggered fashion.

While the hardware design of neuronal and synaptic circuits involves a variety of tradeoffs in area, power, reliability, and performance, the more challenging aspect of neuromorphic systems is supporting arbitrary connectivity patterns between spiking neurons. Since the computational advantage of NNs emerges from the high fan-out yet sparse connectivity of neurons, hardware designs also need to support this crucial aspect. Furthermore, several forms of structural plasticity are observed in the brain where neurons can form (or remove) new synaptic connections based on activity. In the following section, we describe some of the architectural design choices that enable the realization of some of these aspects in silicon substrates.

System design principles and approaches

Colocation of memory and computation mitigates the von Neumann bottleneck in neuromorphic processors. Thus, inspired by neuroscience, the state-of-the-art architectural framework of SNN accelerators or coprocessors comprises a network of neurosynaptic cores [3], [4] that can efficiently implement scale-out NNs, as shown in Figure 2. Each core consists of a crossbar array, with electronic synapses at each cross-point

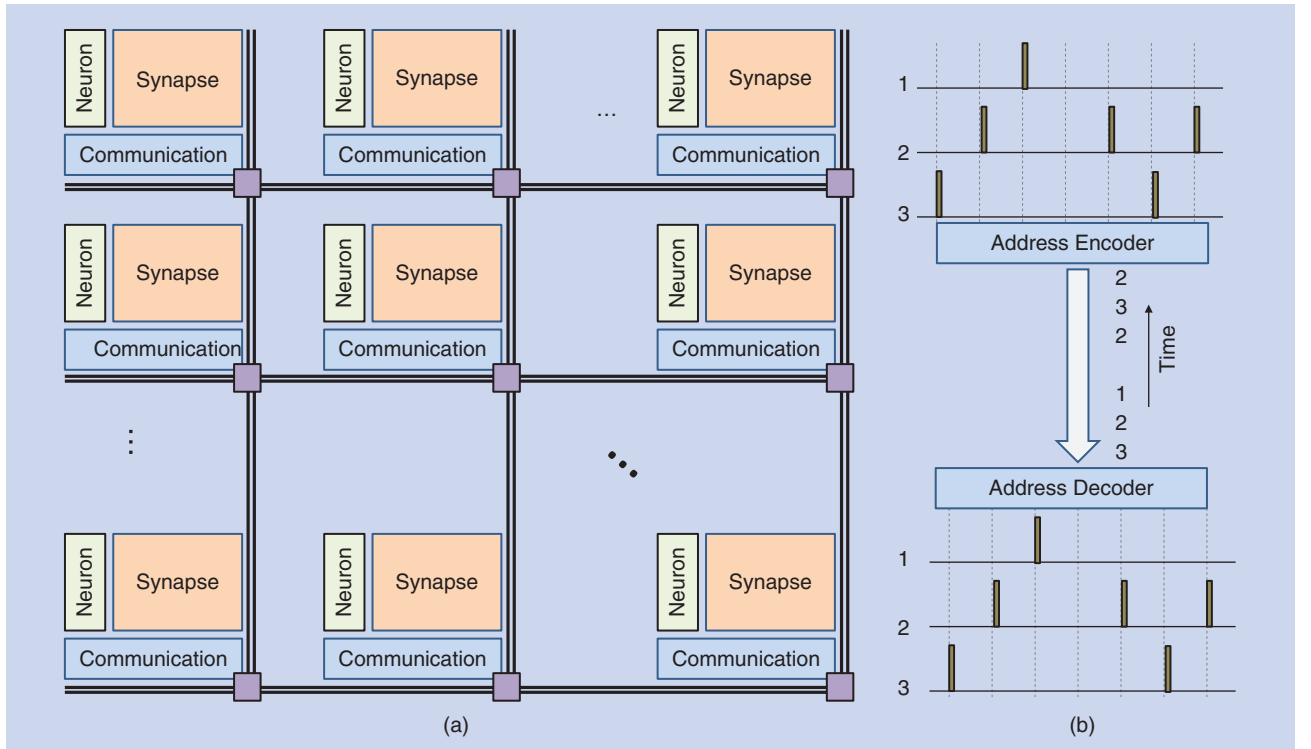


FIGURE 2. The architecture and communication protocols used in neuromorphic systems. (a) A tiled array of neuromorphic cores, with each core integrating neurons and synapses locally. (b) Spikes are communicated between cores through a routing network using address event representation protocols [3], [11].

and peripheral circuitry, including local static random-access memory (SRAM)-based lookup tables for routing information and storage of local data. The peripheral circuitry implements the neuronal functionality (typically the LIF neuron), the read/write electronics, and the driver circuits that control the voltages on the input wires (axons, horizontal lines) and output wires (dendrites, vertical lines). Thus, a neurosynaptic core represents a layer of the SNN, and the passing of information between layers (cores) is enabled by a time-multiplexed communication fabric.

Since neuron spiking rates are orders of magnitude slower than digital electronics, and jitter and delay through digital electronics (propagation and transition delay) are insignificant compared to axonal delays and neuron time constants, the networks typically used to multiplex information from one neurosynaptic core to another are packet-switched networks, using the so-called address event representation (AER) protocol. In this scheme, each neuron has a unique address, which is transmitted to the destination axons when it spikes; the time of spiking is hence encoded implicitly (Figure 2). Note that the AER protocol also allows efficient and flexible off-chip interconnect for large-scale multichip network platforms.

The overall architecture is brain-inspired in the sense that the neurosynaptic core, via its crossbar array, mimics the dense local connectivity of neurons within the same layer, whereas the network interconnect topology with the AER protocol allows sparse long-range connections. From a pragmatic hardware-implementation-oriented viewpoint, such an SNN accelerator architecture with the appropriate core-to-core communications protocol is reminiscent of a dataflow engine, i.e., dataflow from core to core in an asynchronous fashion. See [11] for a general review of neuromorphic design principles.

Although the core-to-core interconnect fabric is implemented with digital CMOS logic, the neurosynaptic core itself can be implemented by using analog/mixed-signal, digital, or memristive technologies. In its simplest form, the neuron membrane potential can be implemented in the analog domain as a voltage across a capacitor or as a multibit variable stored in digital latches. The analog design approaches have focused on two basic design methodologies: subthreshold current-mode circuits and above-threshold circuits. The former approach suffers from higher inhomogeneities (e.g., device mismatch) than the latter one, but it offers lower noise energy (noise power times bandwidth) and better energy efficiency (bandwidth over power) [11].

Alternatively, digital neurons can be realized using CMOS logic circuits, such as adders, multipliers, and counters [14]. From a circuit design point of view, the synapses modulate the input spikes and transform them into a charge that consequently creates postsynaptic currents that are integrated at the membrane capacitance of the postsynaptic neuron [11]. The implementation of silicon synapses typically follows the mixed-signal methodology, although they can also be implemented in the digital domain by employing SRAM cells [14]. Moreover, neurons and synapses can be imple-

mented using memristive technologies, as described in the section “Neuromorphic Computing With Memristive Devices.”

State-of-the-art neuromorphic hardware

In this section, we describe the salient features of state-of-the-art silicon CMOS-based neuromorphic chips. Although research in this domain is more than three decades old, starting with the pioneering work of Carver Mead [1], the discussion in this article is limited to some of the recent demonstrations of large-scale neuromorphic platforms that integrate more than 1,000 neurons.

SpiNNaker

SpiNNaker is a digital system that has been designed to efficiently simulate large spiking networks, approaching the complexity of the brain, in real time [2]. Its building blocks are ARM9 cores that can access a small amount of local memory, plus some additional memory that is shared across one multicore chip. No global memory exists. Nodes can communicate via a high-throughput fabric that is optimized toward routing

small messages (not larger than 72 bits) with high efficiency. SpiNNaker’s event-driven design manifests itself in the message-handling paradigm. A message arriving at a core triggers an interrupt that queues the packet for processing by that core. The system is optimized for small packet handler

code that processes messages quickly, keeping queues short (i.e., not much larger than one). SpiNNaker thus implements fundamental concepts of the brain, such as event-driven computation, locality of information, high fan-in and fan-out connectivity, and communication with tiny messages.

The SpiNNaker system is constructed from processor cores, 18 of which are grouped on a die in a chip. Forty-eight such chips, with up to 864 cores (depending on the manufacturing yield) are assembled on one board. Chips on one board communicate using a custom protocol employing direct connections, wired in a hexagonal topology. Larger systems can be built by connecting 48-chip boards with fast serial interconnects. The largest system in existence today consists of 1 million processors, housed in 10 19-in racks at the University of Manchester, United Kingdom.

While SpiNNaker can be programmed directly, its software stack provides several levels of abstraction, with spiking network implementation being facilitated by PyNN. PyNN is a Python library that supports the portability of network designs between a variety of neuronal simulators and hardware, including the BrainScaleS system (described further on). SpiNNaker’s PyNN interface provides several standard neuron models, such as LIF and Izhikevich’s dynamical systems model, along with common algorithms for synaptic plasticity, including STDP. The successor of this chip, SpiNNaker 2, uses a more modern process technology, increases the number of cores per chip, and adds linear algebra accelerators and several other improvements. It has been used successfully for deep learning with sparse connectivity [15].

Colocation of memory and computation mitigates the von Neumann bottleneck in neuromorphic processors.

TrueNorth

TrueNorth is a million-neuron digital CMOS chip that IBM demonstrated using 28-nm process technology in 2014 [3]. The chip is configured as a tiled array of 4,096 neurosynaptic cores, each containing 12.75 kilobytes of local SRAM memory to store the synapse states, neuron states and parameters, destination addresses of the fan-out neurons, and axonal delays. The digital neuron circuit in each core implements LIF dynamics and is time-multiplexed to emulate the operation of up to 256 neurons, which helps in amortizing the physical area and power budgets. Each core can support the fan-in and fan-out of 256 or less, and this connectivity can be configured such that a neuron in any core can communicate its spikes to one axon in any other core and then to any neuron in that core.

The spike-based communication and routing infrastructure also allows the integration of multiple chips; IBM recently demonstrated the NS16e board that comprises 16 TrueNorth chips. The chip integrates 256 million SRAM synapses, with the synaptic connectivity programmable to two values {0,1}, four programmable 9-bit signed integer weights per neuron, and a programmable spike delivery time at the destination axon in the range of 1–15 time steps. The spike routing is completed asynchronously in every time step, chosen to be 1 ms, to achieve real-time operation akin to biology, although faster synchronization clocks permit accelerated network emulation.

The corelet programming environment is used to map network parameters from software training to the TrueNorth processor. Thanks to the event-driven custom design, the collocation of memory and processing units in each core, and the use of low-leakage silicon CMOS technology, TrueNorth can perform 46 billion synaptic operations per second (SOPS) per watt for real-time operation, with 26 pJ per synaptic event. Its power density of 20 mW/cm² is about three orders of magnitude smaller than that of typical CPUs.

Loihi

Loihi is a neuromorphic learning chip developed by Intel in 2018 using their 14-nm fin field-effect transistor process [4]. This multicore chip supports the implementation of axonal and synaptic delays, neuronal spiking threshold adaptation, and programmable synaptic learning rules based on spike timing and reward modulation. The chip has 128 neural cores, with each core having 1,024 spiking neurons and 2 megabits of SRAM to store the connectivity, configuration, and dynamic state of all neurons within the core. The chip also includes three embedded x86 processors, and 16 megabytes of synaptic memory implemented in SRAM, supporting a synaptic bit resolution of 1–9 bits. Thus, it supports roughly 130,000 neurons and 130 million synapses. Spikes are transported between the cores in the chip using packetized messages by an asynchronous network on chip and allows connection to 4,096 on-chip cores and up to 16,384 chips via hierarchical addressing.

To address the scaling of network connectivity to biological levels (i.e., a fan-out of 1,000), Loihi supports several features, including core-to-core multicast and population-based

hierarchical connectivity. The cores in the chip can be programmed using microcodes to implement several forms of neuromorphic learning rules, such as pairwise STDP, triplet STDP, certain reinforcement learning protocols, and other rules that depend on spike rates as well as spike timing. Under nominal operating conditions, Loihi delivers 30 billion SOPS, consuming about 15 pJ per synaptic operation. A Python package for the Nengo neural simulator allows users to study the implementation of spiking networks on Loihi without accessing the hardware.

BrainScaleS-1

The BrainScaleS system is a mixed-signal platform that combines analog neuron circuits with digital communication [5]. Its unique feature is a speedup factor of 10^3 – 10^4 for spiking network emulations, meaning that a network model running for 1 s of wall time on the hardware emulates up to 10,000 s of biological simulation time. BrainScaleS supports the adaptive exponential I&F model that can be parameterized to exhibit diverse firing patterns. The BrainScaleS system's smallest unit of silicon is the High-Input Count Analog Neuronal Network (HiCANN) chip [16]. The number of neurons per chip can be configured within wide limits, following a tradeoff between the number of input synapses and the number of neurons. A single chip supports a maximum of 512 spiking neurons and up to about 14,000 synapses per neuron. Larger networks are supported by wafer-scale integration of HiCANN chips, which are wired directly on the silicon wafer without cutting it into discrete elements. A single wafer supports 4×10^7 synapses and up to 180,000 neurons.

Prototypes of the next BrainScaleS generation support programmable plasticity via general-purpose processors embedded on the die alongside the neuromorphic circuitry [17]. These processors have access to dedicated sensors at the synapses that measure the time interval between pre- and postsynaptic spikes. Arbitrary functions can be defined that compute updates to the synaptic weights from this information. This enables highly flexible learning rules to be implemented on BrainScaleS, including reward-based learning. BrainScaleS, like SpiNNaker, leverages the PyNN application programming interface (<http://neuralensemble.org/PyNN>) to allow the user to specify spiking networks for emulation on the hardware.

NeuroGrid/Braindrop

The goal of the NeuroGrid platform is to implement large-scale neural models and to emulate their function in real time [6]. Hence, the system's memory and computing resources have time constants that are well matched to the signals that need to be processed. NeuroGrid employs analog/digital mixed-signal subthreshold circuits to model continuous-time neural processing elements. The physics of field-effect transistors operating in the subthreshold regime is used to directly emulate the various neuronal and synaptic functions.

NeuroGrid comprises a board with 16 standard CMOS NeuroCore chips connected in a tree network, with each chip

consisting of a 256×256 array of two compartmental neurons. Each neuron in the array can target multiple destinations by virtue of an asynchronous multicast tree routing digital infrastructure. The full NeuroGrid board can implement models of cortical networks of up to 1 million neurons and billions of synaptic connections with sparse long-range connections and dense local connectivity. A mixed-signal neurosynaptic core called *Braindrop*, with 4,096 neurons and 64 kilobytes of weight memory, has also been recently demonstrated, leveraging the variability of analog circuits for performing useful computations [18].

DYNAP

Dynamic neuromorphic asynchronous processors (DYNAP) is a family of mixed-signal neuromorphic chips from INI Zürich. DYNAP-SE is one such chip, fabricated in 180-nm CMOS technology, integrating 1,024 neurons and 64,000 synapses [7]. The chip is organized into four cores, each having 256 analog neurons. The temporal dynamics of the neurons are implemented using ultralow-power (subthreshold) analog circuits, whereas the asynchronous digital circuits allow for programming and reprogramming the network connectivity at runtime, enabling the configuration of recurrent networks, multilayer networks, and any arbitrary network topology. The analog circuits implement a wide range of features, including multiple types of synaptic and neural dynamics and the spike-frequency adaptation mechanisms that have recently been shown to be crucial in implementing long short-term memory (LSTM)-like networks with spiking neurons.

The asynchronous digital circuits implement a hierarchical routing scheme that combines the best features of all previously proposed approaches (e.g., NeuroGrid's tree-based method and SpiNNaker's 2D-grid mesh scheme), minimizing

the memory requirements. Moreover, the device mismatch that is present in the analog circuits is exploited to implement neural sampling and reservoir-computing strategies that require variability.

ODIN

ODIN is a 28-nm digital neuromorphic chip demonstrated by Catholic University Louvain in 2019 supporting simple forms of on-chip spike-driven synaptic plasticity [8]. The core supports 256 neurons that can be configured to implement first-order LIF dynamics as well as second-order Izhikevich dynamics. The neuronal parameters are stored in a 4-kilobyte SRAM array, and a global controller is used to time-multiplex the neuron logic circuit to implement the dynamics of the neurons in a sequential fashion. The core also integrates 3-bit 256^2 synapses, which are implemented as a 32-kilobyte SRAM array. An additional bit is used per synapse to enable or disable online learning. Using a subset of preprocessed images from the Modified National Institute of Standards and Technology database (commonly known as MNIST) data set, the chip demonstrated on-chip learning achieving 84.5% accuracy and consuming 15 nJ per inference with rank-order coding.

Table 1 summarizes today's state-of-the-art neuromorphic chips, along with some of their key attributes. Note that representative numbers are reported in the table, and, in some instances, it is possible to exceed the performance metrics quoted here by operating the chip at higher frequencies or under other operating conditions. Furthermore, newer-generation prototype chips that form the building blocks of the larger systems have recently been reported, as alluded to in the text (especially for BrainScaleS and SpiNNaker), although we have included the specifications of the full-scale systems in this table.

Table 1. A comparison of state-of-the-art neuromorphic chips, along with some performance attributes.

| Chip | Technology | Integration Density | Key Functionality/Performance Metrics |
|-------------|--------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SpiNNaker | ARM968, 130-nm CMOS (next-generation prototypes: ARM M4F, 28-nm CMOS) | Up to 1,000 neurons/core, 1 million cores | Programmable numerical simulations with 72-bit messages, for real-time simulation of spiking networks |
| TrueNorth | Digital ASIC at 28-nm CMOS | 1 million neurons, 256 million synapses; 1-bit synaptic state to represent a connection, with four programmable 9-bit weights per neuron | SNN emulation without on-chip learning; 26 pJ per synaptic operation |
| Loihi | Digital ASIC at 14-nm CMOS | 130,000 neurons, 130 million synapses with variable weight resolution (1–9 bits) | Supports on-chip learning with plasticity rules, such as Hebbian, pairwise, and triplet STDP, 23.6 pJ per synaptic operation (at nominal operating conditions) |
| BrainScaleS | Mixed-signal wafer-scale system, 180-nm CMOS (next-generation prototype: 65-nm CMOS) | 180,000 neurons, 40 million synapses per wafer | 10^3 – 10^4 -fold acceleration of spiking network emulations, with hardware-supported synaptic plasticity; next-generation prototype: programmable plasticity |
| Braindrop | Mixed-signal 28-nm CMOS | 4,096 neurons, 64,000 programmable weights (with analog circuits that allow realization of all-to-all connectivity) | 0.38 pJ per synaptic update, implements the single core of a planned million-neuron chip |
| DYNAP-SE | Mixed-signal 180-nm CMOS | 1,024 neurons, 64,000 synapses (12-bit content-addressable memory) | Hybrid analog/digital circuits for emulating synapse and neuron dynamics, 17 pJ per synaptic operation |
| ODIN | Digital ASIC at 28-nm CMOS | 256 neurons, 64,000 synapses with 3-bit weight and 1 bit to encode learning | 12.7 pJ per synaptic operation, implements on-chip spike-driven plasticity |

Neuromorphic computing with memristive devices

Going beyond conventional CMOS, a new class of emerging nanoscale devices, namely, resistive memory, or memristive devices with their nonvolatile storage capability, is particularly well suited for developing computing substrates for SNNs. In these devices, information is stored in their resistance or conductance states. The four main types of memristive devices are phase-change memory (PCM), metal–oxide-based resistive RAM (RRAM), conductive bridge RAM (CBRAM), and spin-transfer-torque magnetic RAM (STT-MRAM) [see Figure 3(a)]. The resistance values of these devices are altered by the application of appropriate electrical pulses through various physical mechanisms, such as phase transition, ionic drift, and spintronic effects. Besides this ability to achieve multiple resistance values, many of these devices also exhibit an accumulative behavior whereby the resistance values can be incrementally increased or decreased upon the application of successive programming pulses of the same amplitude. These attributes are key to their application in neuromorphic computing, as illustrated in Figure 3(b) for PCM devices.

The accumulative behavior of memristive devices can be exploited to emulate neuronal dynamics [10]. In one approach using PCM devices, the internal state of the neuron is represented by the phase configuration of the device [Figure 4(a)]. By translating the neuronal input to appropriate electrical signals, the firing frequency can be tuned in a highly controllable manner proportional to the strength of the input signal.

In addition to the deterministic neuronal dynamics, stochastic neuronal dynamics also play a key role in signal encoding

The accumulative behavior of memristive devices can be exploited to emulate neuronal dynamics.

and transmission in biological NNs. One notable example is the use of neuronal populations to represent and transmit sensory and motor signals. The PCM-based neurons exhibit significant interneuronal as well as intraneuronal randomness, thus mimicking

this stochastic neuronal behavior at the device level. Hence, multiple I&F cycles in a single phase-change neuron could generate a distribution of interspike intervals, and this enables population-based computation. By exploiting this, fast signals were shown to be accurately represented by population coding, despite the rather slow firing rate of the individual neurons [Figure 4(b)].

Memristive devices organized in a crossbar architecture can also be used to emulate the two essential synaptic attributes, namely, synaptic efficacy and plasticity (Figure 5). Synaptic efficacy refers to the generation of a synaptic output based on the incoming neuronal activation; this can be realized using Ohm's law by measuring the current that flows through the device when an appropriate read voltage signal is applied. Synaptic plasticity, in contrast, is the ability of the synapse to change its weight, typically during the execution of a learning algorithm. The crossbar architecture is well suited to implement synaptic plasticity in a parallel and efficient manner by the application of suitable write pulses along the wires of the crossbar.

Although nanoscale devices offer exciting computational possibilities and scaling potential, several challenges need to be overcome to enable commercial products. PCM is based on the rapid and reversible phase transition of certain types of materials, such as germanium antimony telluride ($\text{Ge}_2\text{Sb}_2\text{Te}_5$). However, it is necessary to reduce the programming

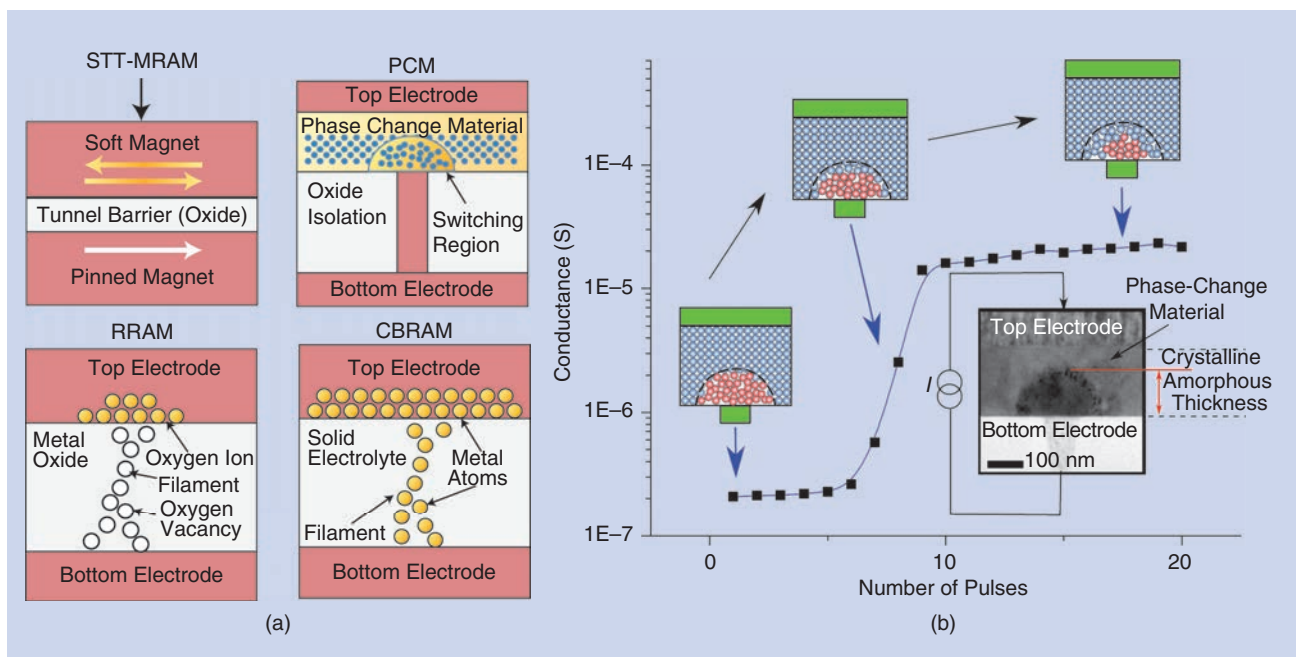


FIGURE 3. (a) A schematic illustration of memristive devices: STT-MRAM, PCM, RRAM, and CBRAM [19]. (b) The incremental programming of PCM conductance [10].

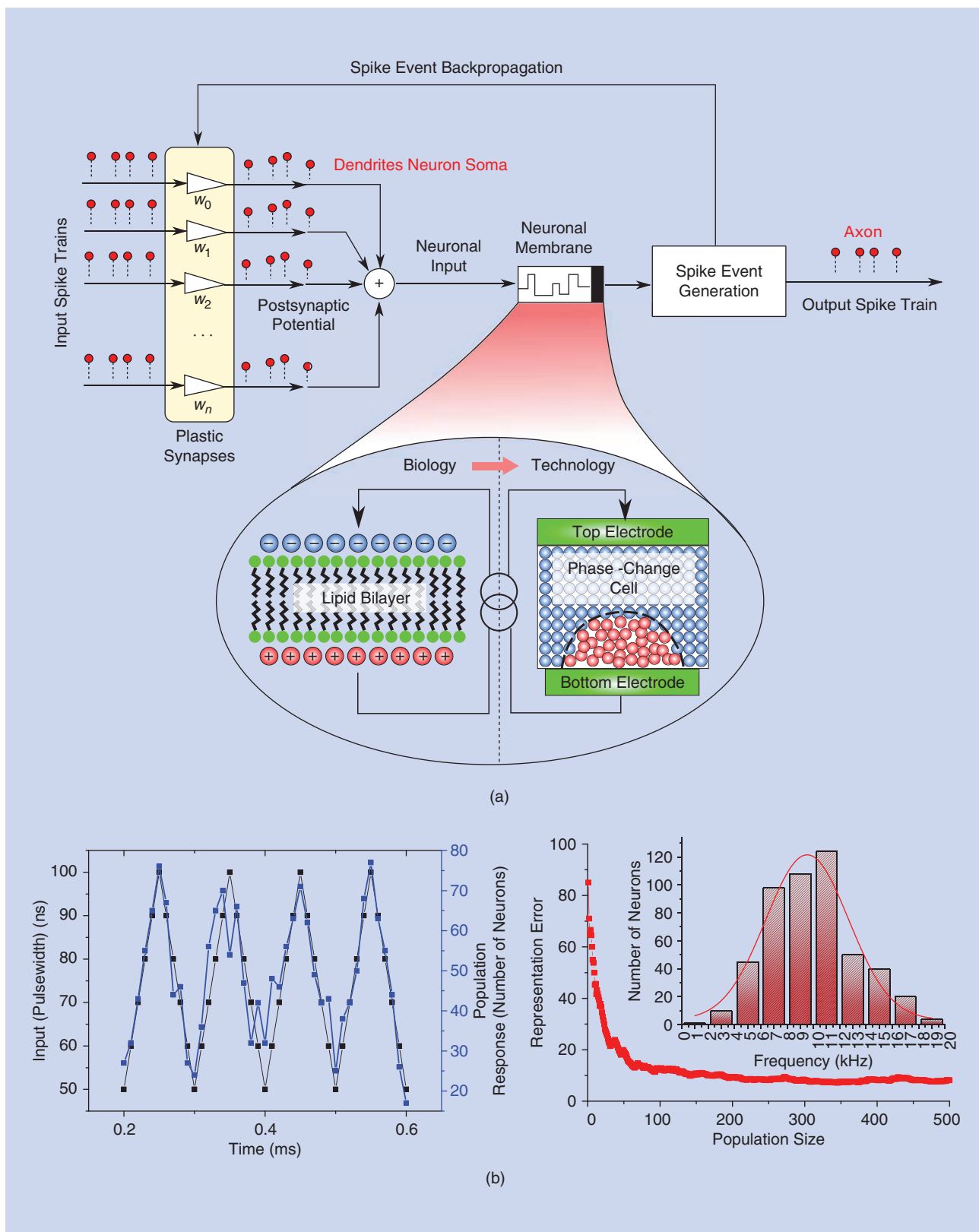


FIGURE 4. (a) A schematic illustration of a phase-change neuron that consists of the dendrites, the soma, and the axon. The key element is the neuronal membrane, which stores the membrane potential in the phase configuration of a PCM device. It is possible to connect the dendrites to plastic synapses that interface the neuron with other neurons in a network. (b) The representation of high-frequency signals via population coding of 500 slow-firing stochastic phase-change neurons. Also shown is the error in the representation of the stimulus by the population code. The population code captures the input signal despite all of the neurons in the population having their actual spiking frequency less than twice the base frequency of the input [10].

current as well as improve the temporal stability of the achieved conductance states. RRAM and CBRAM typically rely on the formation and rupture of nanoscale filaments to achieve multiple conductance values. This filamentary mechanism is particularly prone to inter- and intradevice variations, which is currently the major technical hurdle.

STT-MRAM devices consist of two magnetic layers separated by a tunnel barrier. These devices exhibit two resistive states, depending on whether the magnetization of the two layers is in a parallel or antiparallel direction. Devices based on STT-MRAM are expected to exhibit almost unlimited endurance and faster switching compared to those involving RRAM and PCM. However, the key challenge is a substantially lower dynamic range in programmable conductance states (typically a factor of two to three) as opposed to PCM and RRAM (which exhibit a dynamic range exceeding 100). It is crucial that new circuits and archi-

it is expected that significant gains in area and power efficiency are possible by employing nanoscale memristive devices in neuromorphic processors.

tectures are developed that can mitigate these nonidealities [21].

There are also circuit-level challenges, such as the voltage drop across the long wires connecting the devices as well as the overhead introduced by data converters and other peripheral circuitry. These aspects would limit the size of the memristive crossbars that could be realized. However, in spite of these challenges, it is expected

that significant gains in area and power efficiency are possible by employing nanoscale memristive devices in neuromorphic processors [22].

Signal processing applications

Neuromorphic processors strive to balance the efficiency of computation with the energy needed for this computation, similar to the human brain. Systems enabled by such processors are expected to have the first impact on smart edge devices, such as wearables, mobile devices, Internet of Things (IoT)

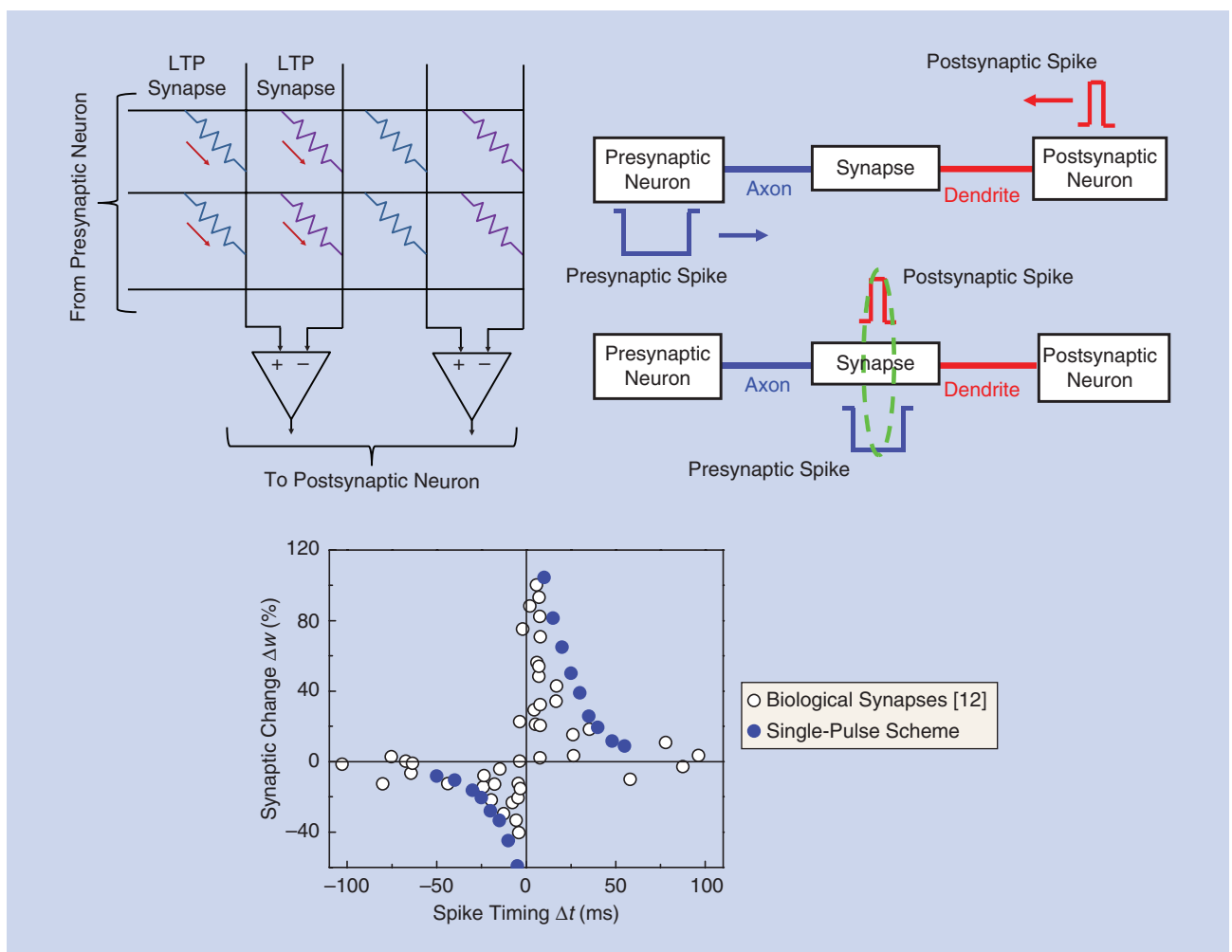


FIGURE 5. Memristive devices organized in a crossbar configuration can be used to emulate synaptic communication and plasticity behaviors, such as STDP, by applying programming pulses from the periphery. Two devices, one for potentiation (LTP) and the other for depression (LTD), are used to represent the synapse in the array [20]. (Adapted from [20].)

sensors, and driverless vehicles, which have stringent constraints on size, weight, area, and power and are required to intelligently interact with the world autonomously for extended periods of time. Neuromorphic approaches could result in highly power-efficient devices capable of responding quickly in an intelligent manner in dynamic environments for such applications.

An illustrative example of this neuromorphic approach is the recent demonstration of a recurrent SNN that learns directly from data streams while using only a limited number of training examples and a relatively small memory for learning [24]. Real-world data, represented in the form of spikes, were used for event-triggered learning, which was implemented in a microprocessor operating in the asynchronous mode [23]. SNNs naturally enable sparse representations of data and can be efficiently implemented using asynchronous logic because data processing occurs only during spike events (Figure 6). By implementing local learning rules on networks with sparse connectivity, both the memory required to store the network parameters and the time needed to train the model can be minimized significantly compared to traditional machine-learning techniques [24]. This approach that implements SNNs in asynchronous processors has huge potential to enable edge devices that can learn and infer efficiently in the field.

Along with the development of neuromorphic processors, bioinspired event-driven sensors have emerged, which can further accelerate the development of intelligent edge devices [11]. The most notable among them is the dynamic vision sensor (DVS) camera, inspired by the information processing mechanisms of the retina, and the silicon cochlea chip, inspired by how the inner ear encodes sound signals in the spike domain. These sensors have superior performance compared to conventional sensors in several respects. For instance, the DVS camera has a 1,000-times greater advantage in sampling rate compared to a conventional camera, which helps in capturing fast-changing events in the visual scene [25]. In recent years, there have been several demonstrations of systems that combine such event-driven sensors with efficient neuromorphic processors, some of which will be discussed subsequently.

We now describe some of the proof-of-concept demonstrations targeting signal processing applications using the neuromorphic platforms discussed in the section “State-of-the-Art Neuromorphic Hardware.” In a notable example using IBM’s TrueNorth chip,

Neuromorphic approaches could result in highly power-efficient devices capable of responding quickly in an intelligent manner in dynamic environments.

deep NNs were trained with a modified backpropagation rule, so that the weights and neuronal dynamics could be easily ported to the hardware, which supported only low-precision synaptic weights; software equivalent performance was achieved for several benchmark pattern classification tasks with this approach [26]. In another instance, a convolutional network running on TrueNorth that received video input

from a DVS camera was able to identify the onset of hand gestures with a latency of 105 ms while consuming less than 200 mW [27]. Intel’s Loihi has demonstrated an improvement of more than three orders of magnitude in energy-delay product compared to conventional solvers running on a CPU for least absolute shrinkage and selection operator (commonly known as *LASSO*) optimization problems by using a spiking convolutional implementation of the locally competitive algorithm [4].

Similarly, the SpiNNaker platform has already been used in several applications of spiking networks. A large spiking model of the visual cortex from a neural simulator running on a conventional compute cluster was recently ported to SpiNNaker, demonstrating comparable accuracy and favorable speed and power consumption [28]. Several neuromorphic applications have also been pioneered on the BrainScaleS system, in particular on Spikey, the predecessor of the HiCANN chip. These include several networks performing various computational neuroscience tasks [29] and the first published assessment of pattern recognition on neuromorphic hardware [30].

Because of its ability to operate in real time with biological neuronal signals, NeuroGrid has been used in a closed-loop

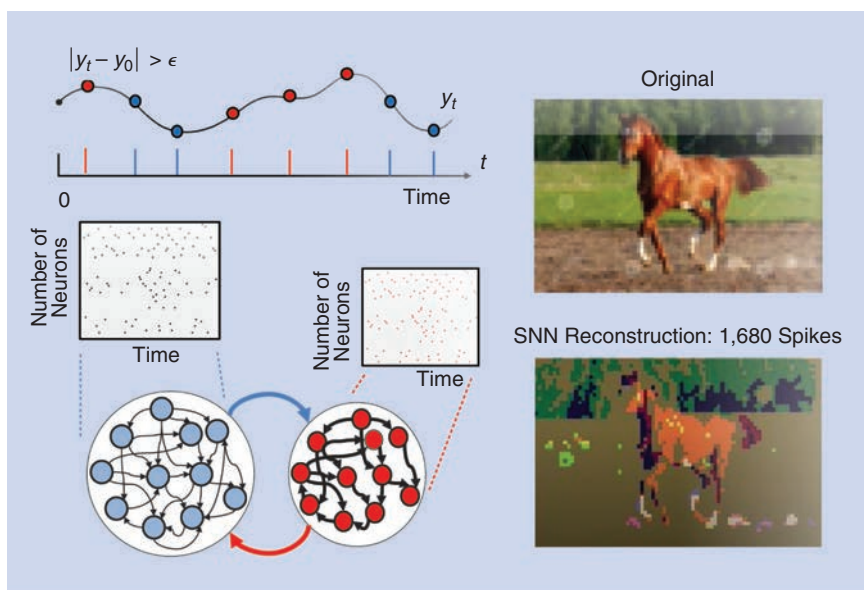


FIGURE 6. (a) Sparse, event-triggered signal encoding, which can be mapped efficiently to recurrent populations of spiking neurons with balanced spiking activity, i.e., between excitatory (blue) and inhibitory (red) neurons, can enable efficient learning. (b) In an illustrative example with this approach, fewer than 2,000 spikes are needed to learn the features of an image that is conventionally represented using more than 100,000 pixels [23].

brain-machine interface (BMI) application [31]. A Kalman-filter-based decoder was implemented via an SNN and tested in BMI experiments with a rhesus monkey. The success of this closed-loop decoder shows the promise of neuromorphic chips for implementing signal processing algorithms in a power-efficient manner, which is a major enabling factor for the clinical translation of neural motor prostheses. The DYNAP-SE chip was recently used for reservoir computing that presented the first steps toward the design of a neuromorphic event-based neural processing system that can be directly interfaced to surface electromyography sensors for the online classification of motor neuron output activities [32].

In spite of being saddled with reliability and variability issues, memristive synapses have also demonstrated immense potential for signal processing applications. One noteworthy example was the demonstration of an SNN for detecting

Bioinspired event-driven sensors have emerged, which can further accelerate the development of intelligent edge devices.

temporal correlations in an unsupervised fashion using plastic PCM synapses [21] (see Figure 7). The network consisted of a spiking neuron receiving an event-based data stream encoded as presynaptic input spikes arriving at PCM synapses. Most of the data streams were temporally uncorrelated, but a small subset was chosen to be mutually correlated. Postsynaptic currents were generated at the synapses that received a spike and were integrated by the neuron, which generated a spike when its membrane potential exceeded a threshold. An STDP rule was used to update the synaptic weights. Since the temporally correlated inputs are more likely to eventually govern the neuronal firing events, the conductance of synapses receiving correlated inputs should increase, whereas that of synapses whose inputs are uncorrelated should decrease. Hence, the final steady-state distribution of the weights should show a separation between synapses receiving correlated and uncorrelated inputs.

In the experiment, 144,000 input streams were fed through more than 1 million PCM devices representing the synapses. As shown in Figure 7(b), well-separated synaptic distributions were achieved in the network at the end of the experiment, even though the devices exhibited significant device-to-device variability and drift in the programmed conductance states. This demonstrated that nanoscale devices can enable complex computational capabilities in neuromorphic hardware platforms.

Future outlook

It is widely believed that because of the added temporal dimension, SNNs should be computationally superior to and thus transcend second-generation deep NNs. The energy efficiency of neuromorphic systems based on SNNs makes them ideal candidates for embedded applications, such as mobile phones, robotics, the IoT, and personalized medicine, which are subject to strict power and area constraints. Moreover, as Moore's law for CMOS scaling is coming to an end, these systems offer unique opportunities to leverage new materials and device structures going beyond standard CMOS processing.

Going forward, there are algorithmic as well as technological challenges. From an algorithmic perspective, despite considerable advances, SNNs are yet to conclusively demonstrate superior

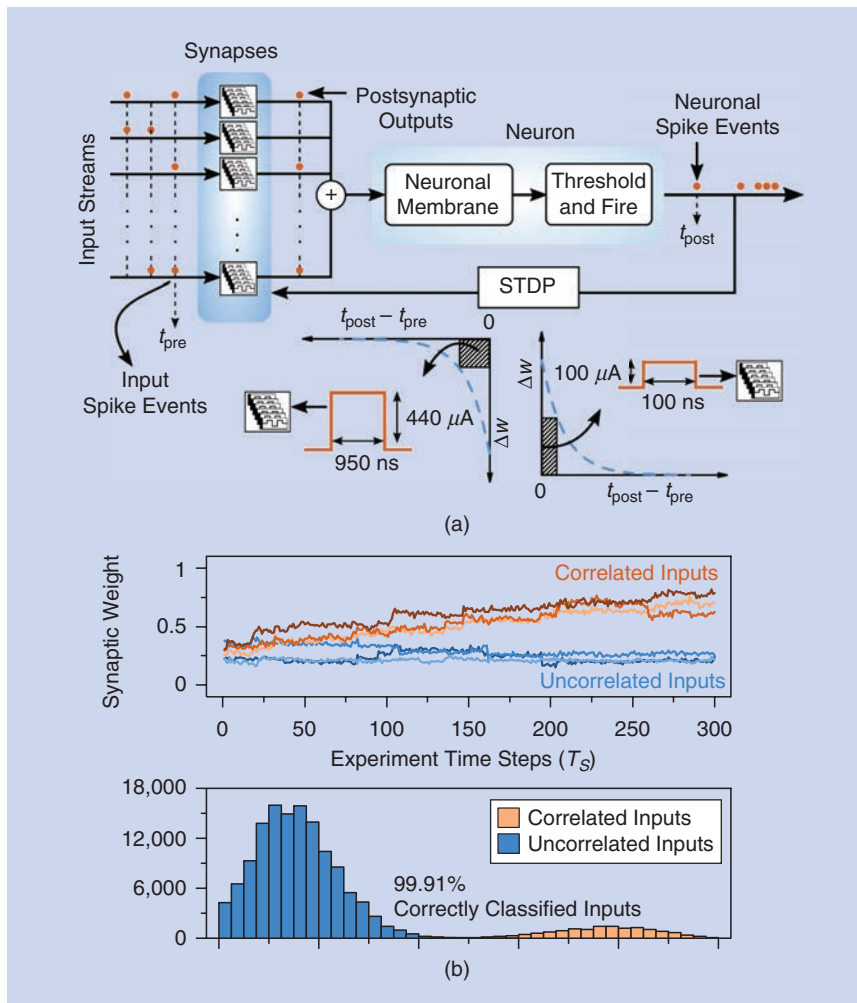


FIGURE 7. (a) An SNN trained to perform the task of temporal correlation detection through unsupervised learning. (b) Synaptic weight evolution as a function of time and the synaptic weight distribution at the end of the experiment. Ten percent of the synapses receive correlated input data streams, with a correlation coefficient of 0.75 [21].

performance compared to conventional deep learning, both in terms of accuracy and in many cases in terms of energy efficiency as well. This gap in performance could be attributed to the lack of efficient and scalable supervised SNN learning algorithms, the lack of efficient local learning rules that can reach the performance of backpropagation, and the reliance on rate coding as opposed to more energy-efficient temporal coding.

Very recently, promising alternatives to rate coding that enable efficient use of spike times have been introduced for SNNs [33]. Moreover, it was shown that recurrent SNNs with adaptive neurons can achieve a classification performance comparable to state-of-the-art LSTM networks [34]. Furthermore, efficient strategies have been demonstrated for converting deep NNs to spiking networks for complex problems, with negligible loss of accuracy [35]. Recently, a novel recurrent artificial NN unit called *Spiking Neural Unit* was introduced that directly incorporates spiking neurons in deep-learning architectures and achieves competitive performance by using backpropagation through time [36].

Finally, it should be noted that some types of neuromorphic hardware support deep learning directly (i.e., without spikes), and powerful algorithms have been developed to leverage the specific advantages of the architectures [15]. Also, modifications of traditional deep-learning algorithms have been proposed that enable their implementation in neuromorphic hardware using binary or ternary representations of neuronal activations and synaptic weights, although higher precision is required for gradient accumulation in these networks during training [37].

From a technology perspective, there are also numerous challenges associated with the use of memristive devices for neuromorphic computing. One key challenge applicable to all memristive technologies is related to the variations in the programmed conductance states with time and ambient temperature. The nonlinearity and stochasticity associated with the accumulative behavior also pose scaling challenges. Recent breakthroughs, such as multicell architectures, hold great promise to address these issues [21].

In summary, we believe that there will be two stages of innovations for the field of low-power, brain-inspired computing platforms. The near-term breakthroughs will come from neuromorphic accelerators built with conventional low-power, mixed-signal CMOS architectures, which are expected to lead to a period of transformative growth involving large neuromorphic platforms designed using ultralow-power computational memories that leverage nanoscale memristive technologies. However, it has to be emphasized that algorithmic exploration has to go hand-in-hand with advances in the hardware technologies.

Acknowledgments

Bipin Rajendran received partial support for this work from the U.S. National Science Foundation grant 1710009 and

Despite considerable advances, SNNs are yet to conclusively demonstrate superior performance compared to conventional deep learning.

grant 2717.001 from the Semiconductor Research Corporation. Michael Schmuker received funding from the European Commission (H2020, Human Brain Project), grant 785907. Abu Sebastian acknowledges support from the European Research Council through the European Union's Horizon 2020 Research and Innovation

Program under grant 682675.

Authors

Bipin Rajendran (bipin@njit.edu) received his B.Tech degree in instrumentation engineering from the Indian Institute of Technology Kharagpur in 2000 and his M.S and Ph.D. degrees in electrical engineering from Stanford University, California, in 2003 and 2006, respectively. He is an associate professor of electrical and computer engineering at the New Jersey Institute of Technology, Newark. Previously, he was a master inventor and research staff member at IBM's T.J. Watson Research Center, Yorktown Heights, New York (2006–2012), and a faculty member in the Department of Electrical Engineering, Indian Institute of Technology, Bombay (2012–2015). His research focuses on building algorithms, devices, and systems for brain-inspired computing. He has authored more than 70 papers in peer-reviewed journals and conferences and has been issued 59 U.S. patents. He is a senior member of the National Academy of Inventors and received an IBM Faculty Award in 2019.

Abu Sebastian (ZRLASE@ch.ibm.com) received his B. E. (honors) degree in electrical and electronics engineering from the Birla Institute of Technology and Science, Pilani, India, in 1998 and his M.S. and Ph.D. degrees in electrical engineering from Iowa State University, Ames, in 1999 and 2004, respectively. He is a principal research staff member and master inventor at IBM Research Zürich. He was a contributor to several key projects in the area of storage and memory technologies and currently leads the research effort on in-memory computing at IBM Research Zürich. He is a corecipient of the 2009 IEEE Control Systems Technology Award and the 2009 IEEE Transactions on Control Systems Technology Outstanding Paper Award. In 2015, he received a European Research Council Consolidator Grant.

Michael Schmuker (m.schmuker@herts.ac.uk) received his M.Sc./diploma degree in biology from Albert Ludwigs University, Freiburg, Germany, in 2003 and his Ph.D. degree in chemistry from Goethe University Frankfurt am Main, Germany, in 2007. He has postdoctoral experience in computational neuroscience and neuromorphic computing and is currently a reader in data science in the Department of Computer Science, University of Hertfordshire, Hatfield, United Kingdom. His research translates neurobiological principles of sensory computing into algorithms for data processing, inference, and control, with a focus on neuromorphic olfaction and gas-based navigation. In 2014, he joined the University of Sussex, Falmer, United Kingdom, on a

Marie Curie Fellowship (European Commission). In 2016, he joined the University of Hertfordshire.

Narayan Srinivasa (physynapse@gmail.com) received his B.Tech degree from the Indian Institute of Technology, Varanasi, in 1988 and his Ph.D. degree from the University of Florida in Gainesville in 1994, both in mechanical engineering. He was a Beckman postdoctoral fellow with the Human-Computer Intelligent Interaction group at the Beckman Institute in University of Illinois at Urbana-Champaign from 1994 to 1997. He is currently the director of machine intelligence research programs at Intel Labs, Santa Clara, California. Prior to that, he was the chief technology officer at Eta Compute, Westlake Village, California, focusing on the development of ultralow-power artificial intelligence solutions for audio applications. From 2016 to 2017, he was chief scientist and senior principal engineer at Intel Labs. He has authored more than 90 papers in peer-reviewed journals and conferences and has been issued 66 U.S. patents.

Evangelos Eleftheriou (ele@zurich.ibm.com) received a diploma of engineering degree from the University of Patras, Greece, in 1979 and a master of engineering and Ph.D. degrees from Carleton University, Ottawa, Canada, in 1981 and 1985, respectively. He is currently responsible for the neuromorphic computing activities at IBM Research Zürich. He was a corecipient of the 2003 IEEE Communications Society Leonard G. Abraham Prize and the 2005 Technology Award of the Eduard Rhein Foundation. In 2009, he was a corecipient of the IEEE Control Systems Technology Award and the IEEE Transactions on Control Systems Technology Outstanding Paper Award. He was appointed an IBM fellow in 2005 and was inducted into the U.S. National Academy of Engineering as a Foreign Member in 2018. He is a Fellow of the IEEE.

References

- [1] C. Mead, "Neuromorphic electronic systems," *Proc. IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990.
- [2] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The SpiNNaker project," *Proc. IEEE*, vol. 102, no. 5, pp. 652–665, 2014.
- [3] P. A. Merolla et al., "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.
- [4] M. Davies et al., "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.
- [5] K. Meier, "A mixed-signal universal neuromorphic computing system," in *Proc. IEEE Int. Electron Devices Meeting*, 2015, pp. 461–464.
- [6] B. V. Benjamin et al., "NeuroGrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proc. IEEE*, vol. 102, no. 5, pp. 699–716, 2014.
- [7] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)," *IEEE Trans. Biomed. Circuits Syst.*, vol. 12, no. 1, pp. 106–122, 2018.
- [8] C. Frenkel, M. Lefebvre, J. Legat, and D. Bol, "A 0.086-mm² 12.7-pJ/SOP 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm CMOS," *IEEE Trans. Biomed. Circuits Syst.*, vol. 13, no. 1, pp. 145–158, 2019.
- [9] D. Kuzum, S. Yu, and H.-S. P. Wong, "Synaptic electronics: Materials, devices and applications," *Nanotechnology*, vol. 24, no. 38, p. 382001, 2013. doi: 10.1088/0957-4484/24/38/382001.
- [10] T. Tuma, A. Pantazi, M. Le Gallo, A. Sebastian, and E. Eleftheriou, "Stochastic phase-change neurons," *Nature Nanotechnol.*, vol. 11, no. 8, pp. 693–699, 2016.
- [11] S.-C. Liu, T. Delbruck, G. Indiveri, A. M. Whatley, and R. Douglas, *Event-Based Neuromorphic Systems*. Hoboken, NJ: Wiley, 2015.

- [12] G. Q. Bi and M. M. Poo, "Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type," *J. Neurosci.*, vol. 18, no. 24, pp. 10,464–10,472, 1998.
- [13] R. Urbanczik and W. Senn, "Reinforcement learning in populations of spiking neurons," *Nature Neurosci.*, vol. 12, no. 3, pp. 250–252, 2009.
- [14] J. Seo et al., "A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," in *Proc. IEEE Custom Integrated Circuits Conf.*, 2011. doi: 10.1109/CICC.2011.6055293.
- [15] C. Liu et al., "Memory-efficient deep learning on a SpiNNaker 2 prototype," *Front. Neurosci.*, vol. 12, Nov. 2018. doi: 10.3389/fnins.2018.00840.
- [16] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *Proc. IEEE Int. Symp. Circuits and Systems*, 2010, pp. 1947–1950.
- [17] S. Friedmann, J. Schemmel, A. Grübl, A. Hartel, M. Hock, and K. Meier, "Demonstrating hybrid learning in a flexible neuromorphic hardware system," *IEEE Trans. Biomed. Circuits Syst.*, vol. 11, no. 1, pp. 128–142, 2017.
- [18] A. Neckar et al., "Braindrop: A mixed-signal neuromorphic architecture with a dynamical systems-based programming model," *Proc. IEEE*, vol. 107, no. 1, pp. 144–164, 2019.
- [19] H.-S. P. Wong and S. Salahuddin, "Memory leads the way to better computing," *Nature Nanotechnol.*, vol. 10, no. 3, pp. 191–194, Mar. 2015.
- [20] D. Kuzum, R. Jeyasingh, S. Yu, and H.-S. Wong, "Low-energy robust neuromorphic computation using synaptic devices," *IEEE Trans. Electron Devices*, vol. 59, no. 12, pp. 3489–3494, 2012.
- [21] I. Boybat et al., "Neuromorphic computing with multi-memristive synapses," *Nature Commun.*, vol. 9, no. 1, 2018. doi: 10.1038/s41467-018-04933-y.
- [22] B. Rajendran, Y. Liu, J. Seo, K. Gopalakrishnan, L. Chang, D. Friedman, and M. Ritter, "Specifications of nanoscale devices & circuits for neuromorphic computational systems," *IEEE Trans. Electron Devices*, vol. 60, no. 1, pp. 246–253, 2013.
- [23] N. Srinivasa and G. Raghavan, "Micropowered intelligence for edge devices," *Embedded Computing Des.*, 2018. [Online]. Available: <https://www.embedded-computing.com/guest-blogs/micropower-intelligence-for-edge-devices>
- [24] P. Panda and N. Srinivasa, "Learning to recognize actions from limited training examples using a recurrent spiking neural model," *Front. Neurosci.*, vol. 3, Mar. 2, 2018. doi: 10.3389/fnins.2018.00126.
- [25] E. Mueggler, B. Huber, and D. Scaramuzza, "Event-based, 6-DOF pose tracking for high-speed maneuvers," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2014, pp. 2761–2768.
- [26] S. K. Esser et al., "Convolutional networks for fast, energy-efficient neuromorphic computing," *Proc. Nat. Acad. Sci.*, vol. 113, no. 41, pp. 11,441–11,446, 2016.
- [27] A. Amir et al., "A low power, fully event-based gesture recognition system," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2017, pp. 7388–7397.
- [28] S. J. van Albada et al., "Performance comparison of the digital neuromorphic hardware SpiNNaker and the neural network simulation software NEST for a full-scale cortical microcircuit model," *Front. Neurosci.*, vol. 12, May 2018. doi: 10.3389/fnins.2018.00291.
- [29] T. Pfeil et al., "Six networks on a universal neuromorphic computing substrate," *Front. Neurosci.*, vol. 7, Feb. 2013. doi: 10.3389/fnins.2013.00011.
- [30] M. Schmuker, T. Pfeil, and M. P. Nawrot, "A neuromorphic network for generic multivariate data classification," *Proc. Nat. Acad. Sci.*, vol. 111, no. 6, pp. 2081–2086, 2014.
- [31] J. Dethier, P. Nuyujukian, C. Eliasmith, T. C. Stewart, S. A. Elasaad, K. V. Shenoy, and K. A. Boahen, "A brain-machine interface operating with a real-time spiking neural network control algorithm," in *Proc. Advances Neural Information Processing Systems*, 2011, pp. 2213–2221.
- [32] E. Donati et al., "Processing EMG signals using reservoir computing on an event-based neuromorphic system," in *Proc. IEEE Biomedical Circuits and Systems Conf.*, 2018, pp. 1–4. doi: 10.1109/BIOCAS.2018.8584674.
- [33] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "STDP-based spiking deep convolutional neural networks for object recognition," *Neural Netw.*, vol. 99, pp. 56–67, Mar. 2018.
- [34] G. Bellec, D. Salaj, A. Subramoney, R. A. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," in *Proc. Neural Information Processing Systems*, 2018, pp. 787–797.
- [35] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: VGG and residual architectures," *Front. Neurosci.*, vol. 13, Mar. 2019. doi: 10.3389/fnins.2019.00095.
- [36] S. Woźniak, A. Pantazi, T. Bohnstingl, and E. Eleftheriou, Deep networks incorporating spiking neural dynamics. 2018. [Online]. Available: <https://arxiv.org/abs/1812.07040>
- [37] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proc. 28th Int. Conf. Neural Information Processing Systems*, vol. 2, 2015, pp. 3123–3131.



©ISTOCKPHOTO.COM/MF3D

Speech Processing for Digital Home Assistants

Combining signal processing with deep-learning techniques

Once a popular theme of futuristic science fiction or far-fetched technology forecasts, digital home assistants with a spoken language interface have become a ubiquitous commodity today. This success has been made possible by major advancements in signal processing and machine learning for so-called far-field speech recognition, where the commands are spoken at a distance from

the sound-capturing device. The challenges encountered are quite unique and different from many other use cases of automatic speech recognition (ASR). The purpose of this article is to describe, in a way that is amenable to the nonspecialist, the key speech processing algorithms that enable reliable, fully hands-free speech interaction with digital home assistants. These technologies include multichannel acoustic echo cancellation (MAEC), microphone array processing and dereverberation techniques for signal enhancement, reliable wake-up word and end-of-interaction detection, and

Digital Object Identifier 10.1109/MSP.2019.2918706
Date of current version: 29 October 2019

high-quality speech synthesis as well as sophisticated statistical models for speech and language, learned from large amounts of heterogeneous training data. In all of these fields, deep learning (DL) has played a critical role.

Evolution of digital home assistants

In the last several years, the smart speaker has emerged as a rapidly growing new category of consumer electronic devices. Smart speakers are Internet-connected loudspeakers containing a digital assistant that can perform a variety of tasks through a hands-free spoken language interface. In many cases, these devices lack a screen and voice is the only input and output modality. These digital home assistants initially performed a small number of tasks, such as playing music, retrieving the time or weather, setting alarms, and basic home automation. Over time, the capabilities of these systems have grown dramatically, as developers have created third-party “skills” in much the same way that smartphones created an ecosystem of apps.

The success of smart speakers in the marketplace can be largely attributed to advances in all of the constituent technologies that comprise a digital assistant, including the digital signal processing involved in capturing the user’s voice, the speech recognition that turns said voice into text, the natural language understanding that converts the text into a user’s intent, the dialog system that decides how to respond, the natural language generation (NLG) that puts the system’s action into natural language, and finally, the speech synthesis that speaks this response to the user.

In this article, we describe in detail the signal processing and speech technologies that are involved in capturing the user’s voice and converting it to text in the context of digital assistants for smart speakers. We focus on these aspects of the system because they are the ones most different from previous digital assistants, which reside on mobile phones. Unlike smartphones, smart speakers are located at a fixed location in a home environment, and thus need to be capable of performing accurate speech recognition from anywhere in the room. In these environments, the user may

Signal attenuation occurs as the sound propagates from the source to the sensor.

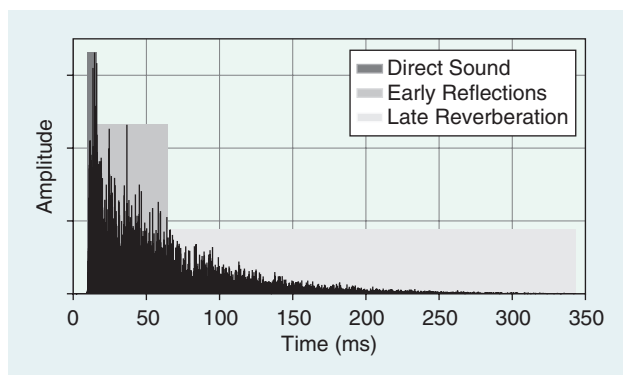


FIGURE 1. An AIR consists of the direct sound, early reflections, and late reverberation.

be several meters from the device; as a result, the captured speech signal can be significantly corrupted by ambient noise and reverberation. In addition, smart speakers are typically screenless devices, so they need to support completely hands-free interaction, including accurate voice activation to wake up the device.

We present breakthroughs in the field of far-field ASR, where reliable recognition is achieved despite significant signal degradations. We show how the DL paradigm has penetrated virtually all components of the system and has played a pivotal role in the success of digital home assistants.

Note that several of the technological advancements described in this article have been inspired or accompanied by efforts in the academic community, which have provided researchers the opportunity to carry out comprehensive evaluations of technologies for far-field robust speech recognition using shared data sets and a common evaluation framework. Notably, the Computational Hearing in Multisource Environments (CHiME) series of challenges [1], [2], the Reverberant Voice Enhancement and Recognition Benchmark (REVERB) Challenge [3], and the Automatic Speech Recognition in Reverberant Environments (ASPIRE) Challenge [4] were met with considerable enthusiasm by the research community.

While these challenges led to significant improvements in the state of the art, they were focused primarily on speech recognition accuracy in far-field conditions as the criterion for success. Factors such as algorithmic latency or computational efficiency were not considered. However, the success of digital assistants in smart speakers can be attributed to not just the system’s accuracy but also its ability to operate with low latency, which creates a positive user experience by responding to the user’s query with an answer shortly after the user stops speaking.

The acoustic environment in the home

In a typical home environment, the distance between the user and the microphones on the smart loudspeaker is on the order of a few meters. There are multiple ways in which this distance negatively impacts the quality of the recorded signal, particularly when compared to a voice signal captured on a mobile phone or headset.

First, signal attenuation occurs as the sound propagates from the source to the sensor. In free space, the power of the signal per unit surface decreases by the square of the distance. This means that if the distance between the speaker and microphone is increased from 2 cm to 1 m, the signal will be attenuated by 34 dB. In reality, the user’s mouth is not an omnidirectional source and, therefore, the attenuation will not be this severe; however, it still results in a significant loss of signal power.

Second, the distance between the source and a sensor in a contained space such as a living room or kitchen causes reverberation as a consequence of multipath propagation.

The wavefront of the speech signal repeatedly reflects off the walls and objects in the room. Thus, the signal recorded at the microphone consists of multiple copies of the source signal, each with a different attenuation and time delay. This effect is described by the acoustic impulse response (AIR) or its equivalent representation in the frequency domain, the acoustic transfer function (ATF). Reverberant speech is thus modeled as the original source signal filtered by the AIR.

An AIR can be broadly divided into direct sound, early reflections (up to roughly the first 50 ms), and late reverberation, as shown in Figure 1. While early reflections are actually known to improve the perceptual quality by increasing the signal level compared to the “dry” direct path signal, the late reverberation causes difficulty in perception—for humans and machines alike—because it smears the signal over time [5].

The degree of reverberation is often measured by the time it takes for the signal power to decrease to -60 dB below its original value; this is referred to as the *reverberation time* and is denoted by T_{60} . Its value depends on the size of the room, the materials comprising the walls, floor, and ceiling, as well as the furniture. A typical value for a living room is between 300 and 700 ms. Because the reverberation time is usually much longer than the typical short-time signal analysis window of 20–64 ms, its effect cannot be adequately described by considering a single speech frame in isolation. Thus, the convolution of the source signal with the AIR cannot be represented by multiplying their corresponding transforms in the short-time Fourier transform (STFT) domain; rather, it is approximated by a convolution over frames.

$$x_{t,f} = \sum_{m=0}^{M-1} a_{m,f} s_{t-m,f}. \quad (1)$$

Here, $x_{t,f}$, $s_{t,f}$, and $a_{t,f}$ are the STFT coefficients of the reverberated signal, source signal, and AIR, respectively, at (discrete) time frame t and frequency bin index f . The length M of the STFT of the AIR is approximately given by T_{60}/B , where B is the frame advance (e.g., 10 ms). Clearly, the effect of reverberation spans multiple consecutive time frames, leading to a temporal dispersion of a speech event over adjacent speech feature vectors.

Third, in a distant-talking speech recognition scenario, it is likely that the microphone will capture other interfering sounds, in addition to the desired speech signal. These sources of acoustic interference can be diverse, hard to predict, and often nonstationary in nature, and thus, difficult to compensate. In a home environment, common sources of interference include TV or radio, home appliances, and other people in the room.

These signal degradations can be observed in Figure 2, which shows signals of the speech utterance “Alexa stop” in 1) a close talk, 2) a distant speech, and 3) a distant speech with additional background speech recording. Keyword detection

and speech recognition are much more challenging in the latter case.

The final major source of signal degradation is the capture of signals that originate from the loudspeaker itself during playback. Because the loudspeaker and the microphones are colocated on the device, the playback signal can be as much as 30–40-dB louder than the user’s voice, rendering the user’s command inaudible if no countermeasures are taken.

System overview

Figure 3 shows the high-level overview of a digital home assistant’s speech processing components. For sound rendering, the loudspeaker system plays music or system responses. For sound capture, digital home assistants employ an array of microphones (typically between two to eight). Due to the form factor of the device, the array is compact with distances between the microphones on the order of a few centimeters. In the following section, techniques from multichannel signal processing are described that can compensate for many of the sources of signal degradation discussed previously.

The signal processing front end performs acoustic echo cancellation, dereverberation, noise reduction (NR), and source separation, all of which aim to clean up the captured signal for input to the downstream speech recognizer. For a true hands-free interface, the system must detect whether speech has been directed to the device. This can be done using

- *wake-word detectors* (also called *hotwords*, *keywords*, or *voice triggers*), which decide whether a user has said the keyword (e.g., “OK Google”) that addresses the device
- *end-of-query detectors*, which are equally important for signaling that the user’s input is complete

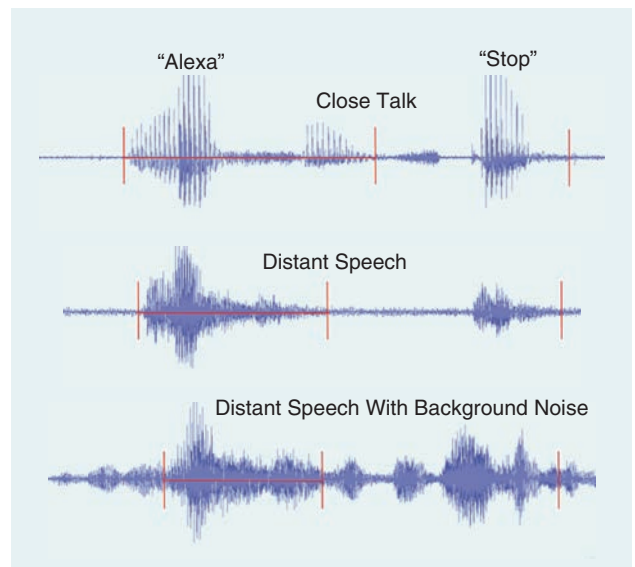


FIGURE 2. A speech utterance starting with the wake word “Alexa” followed by “Stop” in close-talk, reverberated, and noisy, reverberated conditions. The red bars indicate the detected start and end times of the keyword “Alexa” and the end of the utterance.

- *second-turn, device-directed speech classifiers*, which eliminate the need to use the wake word when resuming an ongoing dialogue
- *speaker identification modules*, which make the system capable of interpreting a query in a user-dependent way.

Once device-directed speech is detected, it is forwarded to the ASR component.

The recognized word sequence is then forwarded to the natural language processing (NLP) and dialog management subsystem, which analyzes the user input and decides on a response. The NLG component prepares the desired system response, which is spoken out on the device through the text-to-speech (TTS) component. Note that NLP is beyond the scope of this article. The remainder of this article focuses on the various speech processing tasks.

Some of the aforementioned processing tasks are carried out on the device, typically those close to the input–output, while others are done on the server. Although the division between client and server may vary, it is common practice to run signal enhancement and wake-word detection on the device, while the primary ASR and NLP are done on the server.

Multichannel speech enhancement

The vector of the D microphone signals $\mathbf{y} = (y_1, \dots, y_D)^T$ at time–frequency (tf) bin (t, f) can be written in the STFT domain [6] as

$$\mathbf{y}_{t,f} = \underbrace{\sum_{i=1}^{N_s} \sum_{m=0}^{M-1} \mathbf{a}_{m,f}^{(i)} s_{t-m,f}^{(i)}}_{\text{speech}} + \underbrace{\sum_{j=1}^{N_o} \sum_{m=0}^{M-1} \mathbf{w}_{m,f}^{(j)} o_{t-m,f}^{(j)}}_{\text{playback}} + \underbrace{\mathbf{n}_{t,f}}_{\text{noise}}. \quad (2)$$

The first sum is over the N_s speech sources $s_{t,f}^{(i)}, i = 1, \dots, N_s$, where $\mathbf{a}_{t,f}^{(i)}$ is the vector of ATFs from the i th source to the microphones. The second sum describes the playback of the N_o loudspeaker signals $o_{t,f}^{(j)}, j = 1, \dots, N_o$, which are inadvertently captured by the microphones via the ATF vector $\mathbf{w}_{t,f}^{(j)}$

at frequency bin f . Additionally, $\mathbf{n}_{t,f}$ denotes additive noise; here, we assume for simplicity that the transfer functions are time invariant and of the same length.

It is only one of many signals, which contains the user's command, while all other components of the received signal are distortions. In the following section we describe how to extract this desired signal.

MAEC

MAEC is a signal processing approach that prevents signals generated by a device's loudspeaker from being captured by the device's own microphones and confusing the system. MAEC is a well-established technology that relies on the use of adaptive filters [7]; these filters estimate the acoustic paths between loudspeakers and microphones to identify the part of the microphone signal that is caused by the system output and then subtracts it from the captured microphone signal.

Linear adaptive filters can suppress the echoes by typically 10–20 dB, but they cannot remove them completely. One reason is the presence of nonlinear components in the echo signal, which are caused by loudspeaker nonlinearities and mechanical vibrations. Another reason is that the filter lengths must not be chosen to be too large to enable fast adaptation to changing echo paths. These lengths are usually shorter than the true loudspeaker-to-microphone impulse responses. Furthermore, there is a well-known ambiguity issue with system identification in MAEC [7].

Therefore, it is common practice in acoustic echo cancellation to employ a residual echo suppressor following echo cancellation. In a modern digital home assistant, its filter coefficients are determined with the help of a neural network (NN) [6]. The deep NN (DNN) is trained to estimate, for each tf bin, a speech presence probability (SPP). Details of this procedure are described in “Unsupervised and Supervised Speech Presence Probability Estimation.” From this SPP a mask can be computed,

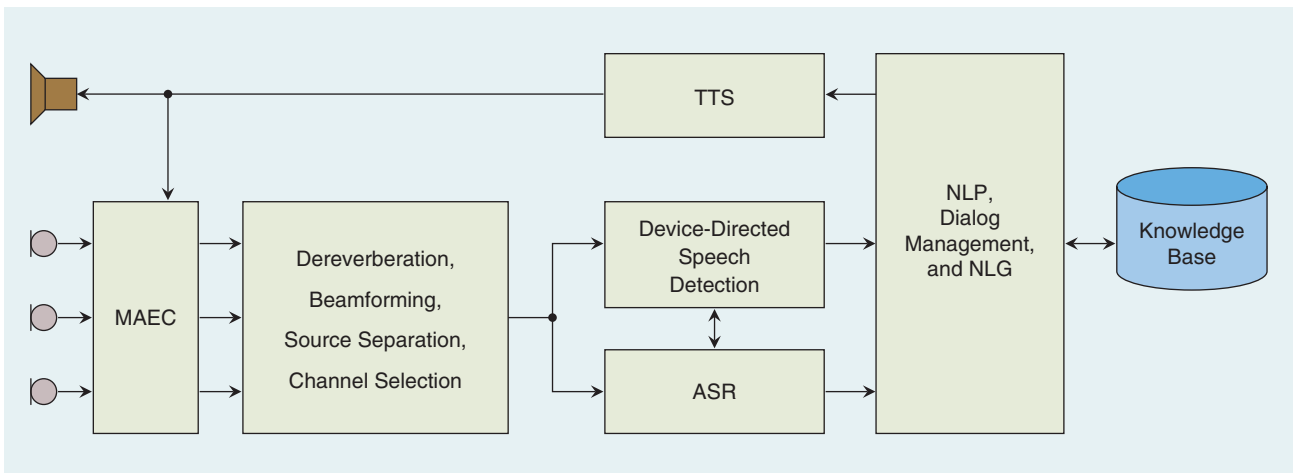


FIGURE 3. An overview of the example architecture of signal processing tasks in a smart loudspeaker.

which separates desired speech-dominated tf bins from those dominated by residual echoes, and from this information, the coefficients of a multichannel filter for residual echo suppression are computed.

With MAEC in place, it is possible that the device can listen to a command, while the loudspeaker is in use, e.g., playing music. The user can barge in and still be understood, an important feature for user convenience. Once the wake-up keyword has been detected, the loudspeaker signal and MAEC are ducked or switched off, while the speech recognizer is activated.

Dereverberation

We now turn our attention to the first sum in (2). Assuming for simplicity that a single speech source is present, this term simplifies to (1).

As mentioned previously, it is the late reverberation that is harmful to speech recognition performance. Decomposing the reverberated signal into the direct sound and early reflections $\mathbf{x}_{t,f}^{(\text{early})}$ and the late reverberation $\mathbf{x}_{t,f}^{(\text{late})}$ according to

$$\mathbf{x}_{t,f} = \mathbf{x}_{t,f}^{(\text{early})} + \mathbf{x}_{t,f}^{(\text{late})}, \quad (3)$$

it is the late reverberation that a dereverberation algorithm aims to remove, while preserving the direct signal and early reflections.

There is a wealth of literature on signal dereverberation [5]. Approaches can be broadly categorized into linear filtering and magnitude or power spectrum-estimation techniques. For ASR tasks, the linear filtering approach is recommended because it does not introduce nonlinear distortions to the signal, which can be detrimental to speech recognition performance.

Using the signal model in (1) where the AIR is a finite impulse response, a Kalman filter can be derived as the statistically optimum linear estimator under a Gaussian source assumption. Because the AIR is unknown and even time varying, the Kalman filter is embedded in an expectation maximization (EM) framework, where Kalman filtering and signal parameter estimation alternate [8].

If the reverberated signal is modeled as an autoregressive stochastic process instead, linear prediction-based dereverberation filters can be derived. A particularly effective method that has found widespread use in far-field speech recognition is the weighted prediction error (WPE) approach [9]. WPE can be formulated as a multiple-input, multiple-output filter,

Unsupervised and Supervised Speech Presence Probability Estimation

In the unsupervised learning approach, a spatial mixture model is used to describe the statistics of $\mathbf{y}_{t,f}$ or a quantity derived from it:

$$p(\mathbf{y}_{t,f}) = \sum_{k=0}^1 \pi_k p(\mathbf{y}_{t,f} | \theta_k), \quad (S1)$$

where we assumed a single speech source and where π_k is the a priori probability that an observation belongs to mixture component k and $p(\mathbf{y}_{t,f} | \theta_k)$ is an appropriate component distribution with parameters θ_k [17]–[19]. This model rests upon the well-known sparsity of speech in the short-time Fourier transform (STFT) domain [20]

$$\mathbf{y}_{t,f} = \begin{cases} \mathbf{a}_f s_{t,f} + \mathbf{n}_{t,f} & z_{t,f} = 1 \\ \mathbf{n}_{t,f} & z_{t,f} = 0 \end{cases} \quad (S2)$$

where $z_{t,f}$ is the hidden class affiliation variable, which indicates speech presence. The model parameters are estimated via the expectation maximization (EM) algorithm, which delivers the speech presence probability (SPP) $\gamma_{t,f} = \Pr\{z_{t,f} = 1 | \mathbf{y}_{t,f}\}$ in the E-step [21].

The supervised learning approach to SPP estimation employs a neural network (NN). Given a set of features extracted from the microphone signals at its input and the true class affiliations $z_{t,f}$ at the output, the network is trained to output the SPP $\gamma_{t,f}$ [22], [23]. Because all of the STFT bins $f = 0, \dots, F-1$ are used as inputs, the network is

able to exploit interfrequency dependencies, while the mixture model-based SPP estimation operates on each frequency independently. If additional cross-channel features, such as interchannel phase differences, are used as inputs, spatial information can also be exploited for SPP estimation.

In a batch implementation, given the SPP the spatial covariance matrices of speech plus noise and noise are estimated by

$$\begin{aligned} \Sigma_f^{(\mathbf{y})} &= \frac{\sum_t \gamma_{t,f} \mathbf{y}_{t,f} \mathbf{y}_{t,f}^H}{\sum_t \gamma_{t,f}}; \\ \Sigma_f^{(\mathbf{n})} &= \frac{\sum_t (1 - \gamma_{t,f}) \mathbf{y}_{t,f} \mathbf{y}_{t,f}^H}{\sum_t (1 - \gamma_{t,f})}. \end{aligned} \quad (S3)$$

From these covariance matrices, the beamformer coefficients of most common beamformers can be readily computed [21]. By an appropriate definition of the noise mask, this concept can also be extended to noisy and reverberant speech, leading to a significant dereverberation effect of the beamformer [24], as shown in Figure 4.

Low latency in a smart loudspeaker is important and impacts both the design of the EM (or statistical methods in general) and the NN-based approaches, see, e.g., [6], [15], and [25] for further discussion.

allowing further multichannel processing, such as beamforming, to follow it [10], [11]. The underlying idea of WPE is to estimate the late reverberation $\mathbf{x}_{t,f}^{(\text{late})}$ and subtract it from the observation to obtain a maximum likelihood estimate of the early arriving speech

$$\hat{\mathbf{x}}_{t,f}^{(\text{early})} = \mathbf{x}_{t,f} - \mathbf{G}_{t,f} \tilde{\mathbf{x}}_{t-\Delta,f}. \quad (4)$$

Here, $\mathbf{G}_{t,f}$ is a matrix containing the linear prediction coefficients for the different channels and $\tilde{\mathbf{x}}_{t-\Delta,f}$ are stacked representations of the observations: $\tilde{\mathbf{x}}_{t-\Delta,f} = (\mathbf{x}_{t-\Delta,f}^T, \dots, \mathbf{x}_{t-\Delta-L+1,f}^T)^T$, where L is the length of the dereverberation filter. It is important to note that $\hat{\mathbf{x}}_{t,f}^{(\text{early})}$ at time frame t is estimated from observations at least Δ frames in the past. This ensures that the dereverberation filter does not destroy the inherent temporal correlation of a speech signal, which is not caused by the reverberation. The filter coefficient matrix cannot be estimated in closed form; the reason is that the driving process of the autoregressive model, $\mathbf{x}_{t,f}^{(\text{early})}$, has an unknown and time-varying variance $\lambda_{t,f}$. However, an iterative procedure can be derived, which alternates between estimating the variance $\lambda_{t,f}$ and the matrix of filter coefficients $\mathbf{G}_{t,f}$ on signal segments.

Because WPE is an iterative algorithm, it is not suitable for use in a digital home assistant, where low latency is important; however, the estimation of the filter coefficients can be cast as a recursive least squares problem [12]. Furthermore, using the average over a window of observed speech power spectra as an estimate of the signal variance $\lambda_{t,f}$, a very efficient low-latency version of the algorithm can be used [13].

Many authors reported that WPE leads to word error rate (WER) reductions of a subsequent speech recognizer [13], [14]. How much of a WER reduction is achieved by dereverberation depends on many factors such as degree of reverberation, signal-to-noise ratio (SNR), difficulty of the ASR task, robustness of the models in the ASR decoder, and so on. In [13], relative WER improvements of 5–10% were reported on simulated digital home assistant data with a pair of microphones and a strong back-end ASR engine.

Multichannel NR and beamforming

Multichannel NR aims to remove additive distortions, denoted by $\mathbf{n}_{t,f}$ in (2). If the AIR from the desired source to the sensors is known, a spatial filter (i.e., a beamformer), can be designed that emphasizes the source signal over signals with different transfer characteristics. In its simplest form, this filter compensates for the different propagation delays that the signals at the individual sensors of the microphone array exhibit and that are caused by their slightly different distances to the source.

For the noisy and reverberant home environment, this approach, however, is too simplistic. The microphone signals differ not only in their relative delay, the whole reflection pattern they are exposed to is different. Assuming again a single speech source and good echo suppression and dereverberation, (2) reduces to

$$\mathbf{y}_{t,f} = \mathbf{x}_{t,f} + \mathbf{n}_{t,f} \approx \mathbf{a}_f s_{t,f} + \mathbf{n}_{t,f}, \quad (5)$$

where \mathbf{a}_f is the vector form of the AIRs to multiple microphones, and where we assume it to be time invariant under the condition that the source and microphone positions do not change during a speech segment (e.g., an utterance). Note that unlike (1) and (2), the multiplicative transfer function approximation is used here, which is justified by the preceding dereverberation component. Any signal component that deviates from this assumption can be viewed as captured by the noise term $\mathbf{n}_{t,f}$. Similarly, residual echoes can be viewed as contributing to $\mathbf{n}_{t,f}$, which results in a spatial filter for denoising, dereverberation, and residual echo suppression.

Looking at (5), it is obvious that $s_{t,f}$ and \mathbf{a}_f can only be identified up to a (complex-valued) scalar because $s_{t,f} \cdot \mathbf{a}_f = (s_{t,f} \cdot C) \cdot (\mathbf{a}_f/C)$. To fix this ambiguity, a scale factor is chosen such that for a given reference channel, e.g., channel 1, the value of the transfer function is 1. This yields the so-called relative transfer function (RTF) vector $\tilde{\mathbf{a}}_f = \mathbf{a}_f/a_{1,f}$.

Spatial filtering for signal enhancement is a classic and well-studied topic for which statistically optimal solutions are known; however, these textbook solutions usually assume that the RTF $\tilde{\mathbf{a}}_f$, or its equivalent in anechoic environments (i.e., the vector of time difference of arrival), are known, which is an unrealistic assumption. The key to spatial filtering is, again, SPP estimation (see “Unsupervised and Supervised Speech Presence Probability Estimation.”) The SPP tells us which tf bins are dominated by the desired speech signal and which are dominated by noise. Given this information, spatial covariance matrices for speech and noise can be estimated, from which, in turn, the beamformer coefficients are computed. An alternative approach is to use the SPP to derive a tf mask, which multiplies tf bins dominated by noise with zero, thus leading to an effective mask-based NR.

Figure 4 shows the effectiveness of beamforming for an example utterance. The spectrogram, i.e., the tf representation of a clean speech signal, is displayed in Figure 4(a), followed in Figure 4(b) by the same utterance after convolution with an AIR, and in Figure 4(c) after the addition of noise. Figure 4(d) shows the output of the beamformer, which effectively removed noise and reverberation.

The usefulness of acoustic beamforming for speech recognition is well documented. On the CHiME 3 and 4 challenge data, acoustic beamforming reduced the WER by nearly half. On typical digital home assistant data, WER reductions on the order of 10–30% relative were reported [6], [15], [16].

Source separation and stream selection

Now we assume that, in addition to the desired speech source, there are other competing talkers, resulting in a total of N_s speech signals, see (2). Blind source separation (BSS) is a technique that can separate multiple audio sources into individual audio streams autonomously. Traditionally, researchers tackle speech source separation using either unsupervised methods, e.g., independent component analysis and

clustering [26], or DL [27], [28]. With clustering in particular, BSS using spatial mixture models is a powerful tool that decomposes the microphone array signal into the individual talkers' signals [17]–[19]. The parameters and variables of those mixture models are learned via the EM algorithm, as explained in “Unsupervised and Supervised Speech Presence Probability Estimation.” The only difference being that the mixture model now has as many components as concurrent speakers. During the EM, for each speaker, a source activity probability (SAP), which is the equivalent to the SPP in the multispeaker case, is estimated.

Extraction of the individual source signals may be achieved using the estimated SAP to derive for each speaker a mask, by which all tf bins not dominated by this speaker are zeroed out. An alternative to this is to use the SAP to compute beamformers, one for each of the speakers, similar to what is explained in “Unsupervised and Supervised Speech Presence Probability Estimation.”

Once the sources are separated, it remains to be decided which of the streams contains the user's command for the digital home assistant. In [6], it is proposed to base this decision on the detection of the wake-up keyword (e.g., “Hey Siri”): If the wake-word detector indicates the presence of the keyword, all streams, i.e., the output streams of source separation and the output of the acoustic beamformer, are scored for the presence of the keyword, and the stream with the highest score is considered to contain the user's command.

ASR

The key knowledge sources for ASR are the acoustic model (AM), the pronunciation model (PM) and the language model (LM). The LM assigns probabilities to hypothesized strings. The PM maps strings to subword units, where the phoneme is a common choice. Probabilities of the acoustic realization of the subword units (generally using some context) are expressed by the AM.

The AM of a speech recognition system is realized by a DNN. Such models estimate the posterior probabilities of subword units in context given the input signal. State-of-the-art network architectures borrow concepts from image recognition networks, e.g., the ResNet [29], and also include sequential modeling through the use of recurrent structures. Many sites use long short-term memory (LSTM) network layers or time-delay NN structures to incorporate that temporal component into the model.

The AM is trained from examples and generally requires large corpora to allow robust parameter estimation of

these models (on the order of thousands of hours). It is essential that these corpora reflect the type of utterances that the device will recognize. For very novel applications, as was the case with the early deployment of digital home assistants, example data was not available, and collecting such large amounts of training data before product launch was considered uneconomical. Complicating matters further, the expected variability is very large for speech coming into such a device; therefore, the bootstrapping problem of a model for a digital home assistant is considerably more complex than for other new application domains.

Bootstrapping the AM

Most sites that developed early digital assistants have large data sets of in-domain, close-talking material available. To make use of that data but render it suitable for the digital home assistant application, simulation techniques are employed. Using the well-known image method [30], sufficiently realistic AIRs can be generated for given room and microphone parameters. Alternatively, measured AIRs can be used, such as the collection in [31]. It is, of course, much easier to simulate thousands of AIRs representing large varieties of room and microphone array configurations in this way than to measure them. The nonreverberant close-talk recordings are then convolved with these AIRs to generate reverberant speech. It should be mentioned, however, that this simulates a static scenario. In reality, an AIR is time-varying,

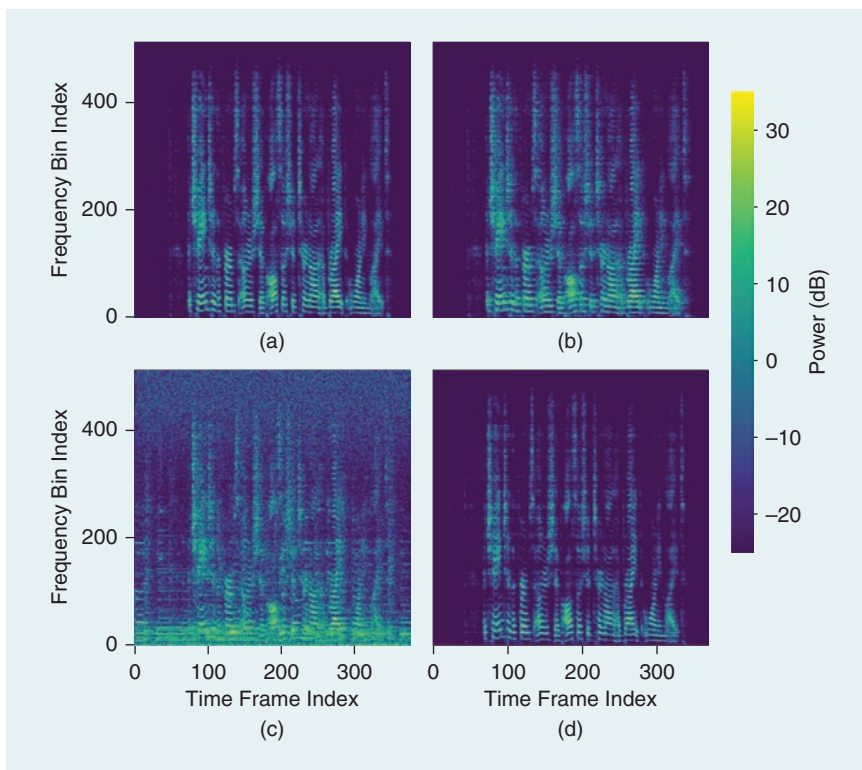


FIGURE 4. The spectrogram of (a) a clean, (b) a reverberated, (c) a noisy and reverberated, and (d) an enhanced speech signal. Enhancement has been achieved with a beamformer that was trained to treat both noise and late reverberation as distortion.

i.e., even the smallest movements of the speaker or changes in the environment will lead to a different reverberation pattern. Nevertheless, experience shows that systems trained with artificially generated reverberant speech perform robustly on real reverberant data.

Additional data augmentation techniques [32] are used in the development of an AM for a new application, which perturb existing recordings along different perceptually relevant parameters, such as speaking rate, vocal tract length, pitch, SNR, noise types, and so forth.

Integrating enhancement and acoustic modeling

Although experimentation showed that the simulation of reverberant distant-talking speech from close-talking corpora was effective in mitigating many of the problems posed in this setting, there is a large body of work that uses enhancement from multichannel processing to (further) mitigate the problems that arise in distant-talking speech recognition, as discussed previously. However, independent optimization of the enhancement component and acoustic modeling component may not lead to performance improvements per se because a mismatch in the training objectives can adversely affect the overall system performance. It appears advantageous to optimize the AM and enhancement component jointly with a criterion close to the ASR objective. This bypasses the signal-related objective functions, such as maximizing the output SNR, which is used in classic beamforming to ensure that the enhancement result benefits the ASR that consumes its output. This direction was first advocated by [33] in Gaussian mixture-based acoustic modeling.

More recently, it was proposed to perform multichannel enhancement jointly with acoustic modeling in a DNN framework [34], [35]. To leverage the differences in the fine time structure of the signals at the different microphones, it is necessary to input the raw time-domain signal or its equivalent complex-valued STFT representation to the network. This is different from standard acoustic modeling, where the time-domain signal is first compressed to feature vector representations, such as logarithmic mel spectra or cepstra, which no longer carry subtle time information. A close look at the filter coefficients learned in the initial layers of the network showed that, indeed, beamformer-like spatial filters could be identified and that frequency resolutions resembling the mel filter bank were found [34].

An alternative to this single large enhancement and acoustic modeling network is to keep enhancement and AM separate and still optimize both jointly toward an ASR-related criterion. It has been shown in [36] how the NN for SPP estimation (see “Unsupervised and Supervised Speech Presence Probability Estimation,”) can be trained from the objective function of the AM by back-propagating the gradient through the AM DNN and the beamforming operation all the way

to the DNN for SPP estimation. Clearly, this can be viewed as one large DNN with fixed, nontrainable signal processing layers in between.

A direct comparison of the fully integrated approach with separate, albeit jointly trained, speech enhancement and acoustic modeling stages on a common ASR task is unknown. Both techniques have been shown to provide significant WER reductions when compared to that of ASR on single-channel inputs, even when many distant-talking examples were used in training. However, what can be stated is that the integrated approach requires more training data because it has to learn the multichannel processing from the data. The approach with a separate beamformer in front of the AM acts as a kind of regularizer, helping the overall system to settle on appropriate local minima of the networks, thus requiring less training data and being computationally less demanding.

Note that the integration can be extended even from subword unit DNN acoustic modeling to end-to-end speech recognition, which allows for the beamforming components to be optimized jointly within the recognition architecture to improve the end-to-end speech recognition objective [37], [38].

In [39], the effect of integrating enhancement and acoustic modeling was reported using a Google Home production system, where relative WER improvement between 8 and 28% was obtained by integrating WPE dereverberation and DNN-based multichannel processing with the AM of the production system.

Language modeling

Language modeling for the assistant is complex due to the ubiquity of applications for which it is used. Taking sample utterances from these interactions for the entire population of users allows us to estimate an LM that covers the domain as a whole. LMs used in the first pass are n-gram models that predict the next word and the sentence end based on a limited history of typically the three or four preceding words. Speech recognition systems often produce an n-best list in the first pass and apply a rescored second pass using a log-linear or neural LM working on the complete sentence.

For an individual user, however, the actual entropy of his/her utterances is more restricted. For instance, if users want to name a contact, they will likely pick a name that is in their contact list and less likely pick a name that is on the contact list of any user. In other words, a statically trained LM is a good fit for the domain but has poor priors when it comes to an individual user. More generally, the context in which an utterance is produced will have an impact on the content of the utterance. Digital assistant systems generally implement this adjustment by biasing, i.e., adjusting the LM probabilities “on the fly” using the current context. The approach proposed in [40] achieves the biasing by boosting selected n-grams in the LM. An alternate method

Probabilities of the acoustic realization of the subword units are expressed by the AM.

used in [41] does an on-the-fly adjustment of the weights in an LM interpolation.

A second aspect resulting from the multitude of use cases is multilinguality and code switching. Multilingual support for utterance-by-utterance language switching is implemented following the approach proposed in [42] by running several speech recognition systems, one per language, in parallel. The best system and hence, language, is chosen after recognition is completed, either solely based on the scores of the language-dependent systems or supported by a language identification component. Code switching within an utterance is usually implemented by adding a small degree of multilinguality directly to the LM, e.g., an Indian–English speech recognition system usually also covers a limited set of common phrases, such as Hindi, Telugu, and so on. A special case for a virtual assistant are catalogs, such as those for supporting a music or shopping domain, where multilingual content is common. For instance, users of an Indian–English system often ask for music or video titles in their native Indic language.

TTS synthesis

Most smart loudspeakers have no screen on which to display information. On these devices, audio is the most natural way of providing responses to users, and TTS synthesis is used to generate spoken responses.

In the back end of these digital home assistants, an NLG module translates raw data into an understandable text in a markup language. A TTS system takes the markup text as its input and renders speech output. It consists of text analysis (front-end) and speech synthesis (back-end) parts. The text analysis component includes a series of NLP modules, such as sentence segmentation, word segmentation, part-of-speech tagging, dictionary lookup, and grapheme-to-phoneme pronunciation conversion. The speech synthesis part is typically a cascade of prosody prediction and waveform generation modules.

In the digital home assistant domain, the text analysis component can access more contextual information than in other domains (e.g., synthesizing speech for a website), because the NLG module can provide it via markup language. For instance, sometimes it is difficult to disambiguate pronunciation of a place name only from a written text; however, the NLG system can access its knowledge base to resolve the ambiguity and provide it via markup. Furthermore, the front end can also incorporate explicit annotations, providing hints about prosody and discourse domain [43]. Such coupling between NLG and TTS modules allows better synthesis in this domain.

In the back end, either an example-based (concatenative) or model-based (generative) approach is used in the waveform generation module. The former finds the best sequence of small waveform units (e.g., half-phone, phone, and diphone level) from a unit database given a target linguistic or the acoustic features derived from an input text. The lat-

ter first learns a mapping function from text to speech by a model, then predicts a speech waveform given a text and the trained model. The concatenative approach is known to 1) require a large amount of speech data from a single speaker, 2) have a large footprint, 3) be computationally less expensive, and 4) have natural segmental quality but sound discontinuous. On the other hand, the generative approach is known to 1) require less trainable data from multiple speakers, 2) have a small footprint, 3) be computationally expensive, and 4) have smooth transitions but achieve relatively poor vocoder quality. As mentioned previously, achieving

low latency is critical for use in digital home assistants. Vendors choose different approaches to synthesize naturally sounding speech with low latency. We discuss two very different solutions in the following to illustrate the range of options.

The Siri Team at Apple developed on-device, DL-guided, hybrid unit selection concatenative TTS systems [43] to achieve these goals. Conventionally, hidden Markov models (HMMs) were used in hybrid unit selection TTS systems. Later, HMMs were replaced by deep- and recurrent-mixture density networks to compute probabilistic acoustic targets and concatenation costs. Multiple levels of optimizations (e.g., long units, preselection, unit pruning, local caching, and parallel computation) enable the system to produce high-quality speech with an acceptable footprint and computational cost. Additionally, as an on-device system, it can synthesize speech without an Internet connection, allowing on-device, low-latency streaming synthesis. Combined with a higher sampling rate (i.e., 22–48 kHz) and better audio compression, the system achieved significant improvements over their conventional system. This Siri DL-based voice has been used since iOS 10.

On the other hand, Google Home uses a server-side generative TTS system to achieve these goals. Because Google's TTS system runs on servers, an Internet connection is essential; however, even on Wi-Fi-connected smart loudspeakers, an Internet connection can be unstable. Audio streaming using an unstable connection causes stuttering within a response. To prevent stuttering, no streaming is used in Google's TTS service; rather, after synthesizing an entire utterance, the server sends the audio to the device. The device then starts playing the audio after receiving the entire response. Although this approach improves user experience, achieving low latency becomes challenging. To achieve high-quality TTS with low latency, Google developed the Parallel WaveNet-based TTS system [44]. Conventional generative TTS systems often synthesized “robotic”-sounding vocoded speech. The introduction of sample-level autoregressive audio generative models, such as WaveNet [45], has drastically improved the system's naturalness; however, it is computationally expensive and difficult to be parallelized due to its autoregressive nature.

Parallel WaveNet introduced probability density distillation, which allows for the training of a parallel feed-forward

A TTS system takes the markup text as its input and renders speech output.

network from an autoregressive network with no significant difference in segmental naturalness. Because of the parallelization-friendly architecture of Parallel WaveNet, by running it on tensor processing units, it achieved a 1,000 \times speed up (i.e., 20 \times faster than real time) relative to the original autoregressive WaveNet, while retaining its ability to synthesize high-fidelity speech samples. This Parallel WaveNet-based voice has been used in Google Assistant since October 2017.

Fully hands-free interaction

Digital home assistants have a completely hands-free voice-controlled interface. This has important and challenging implications for speech processing systems. The first, most obvious one is that the device must always be listening to recognize when it is addressed by a user. There are other challenges as well, which are described in the following section.

Wake-up word detection

To detect whether a user is addressing the device, a wake-up keyword, e.g., “Alexa,” “OK Google,” or “Hey Siri” is defined. If this word is detected, the device concludes that the ensuing speech is meant for it. It is extremely important for user satisfaction that this keyword detection works very reliably, with both very low false alarm and high recall rates. This, however, is not simple, particularly in the face of poor signal quality (see Figure 2). Certainly, long wake-up words are easier to detect than short ones; however, because the keyword acts as the “name” of the device, its choice is influenced by marketing aspects, leaving minimal room for engineering considerations. Another requirement is low latency; i.e., the system must answer as quickly as a human would do. Furthermore, one must bear in mind that the keyword-spotting algorithm runs on the device. This is different from the ASR component, which is server-borne. Therefore, memory footprint and computational load considerations also play an important role [6].

In one approach proposed in [46], voice activity detection (VAD) is used in an initial step to reduce computation so that the search for a keyword is conducted only if speech has been detected. If speech is detected, a sliding window, whose size depends on the length of the keyword, is swept over the data, and a DNN classifier operates on the frames inside the window. In its simplest form, classification is based upon a fully connected DNN, without any time alignment, resulting in significantly lower computational costs and latency compared to that of ASR. Then, max-pooling along the time axis is carried out on the DNN posteriors to arrive at a confidence score for the presence of a keyword.

To improve detection accuracy, convolutional [47], time-delay [48], or recurrent network layers [49] have been proposed, as well as subword modeling of the keyword and background speech using a DNN-HMM architecture [50], all of which aim to exploit the temporal properties of the

input signal for classification. To further reduce false alarm rates, multistage keyword-detection algorithms have been developed, where initial hypotheses are rechecked using cues like keyword duration, individual likelihoods of the phones comprising the keyword, and so forth [50]. This second-stage classifier is again realized by a DNN. The experiments in [50] show that using subword-based background models can reduce false accept rates (FARs) by roughly 37% relative at a fixed false rejection rate (FRR) of 4%. This work also demonstrates the effectiveness of the two-stage approach, which can reduce FARs by up to 67%, relative to a 4% FRR. A different voice trigger detection system was proposed in [51], where robustness and computational efficiency were achieved using a two-pass architecture.

End-of-query detection

Not only must the beginning of device-directed speech be detected, but also quickly and accurately determining when the user has finished speaking to the system must be accomplished. However, speech pauses must not be taken falsely as the end of the query, nor must ambient noise, e.g., a TV heard in the background, be taken as the continuation of an utterance.

From these considerations, it is clear that a VAD can be no more than one source of information about the end of the user query [52]. Another source of information is the ASR decoder itself. Indeed, because the end-of-query detection is carried out on the server, the ASR engine is available for this task, and its acoustic and LM can be leveraged to identify the end of device-directed speech. An indication

of this is whether the active decoder hypotheses indicate the end of sentence, followed by silence frames. Because low latency is important, the decision cannot be postponed until all competing search hypotheses inside the ASR decoder have died out. To achieve a high degree of reliability, it was proposed to average over all active search hypotheses with this property [53]. Yet another cue for the end of the user query is the recognized word sequence. Those sources of information, VAD, ASR decoder search properties, and the one-best word/character hypothesis can be expressed as fixed-length features, which are input to a dedicated end-of-query DNN classifier [54].

Second-turn, device-directed speech classification

For a natural interaction, it is desirable that the system detects whether another query is meant for it, without the user having to repeat the wake-up keyword again (Example: “Hey Cortana, what is the weather today?”; system answer; “And what about tomorrow?”). One approach toward this functionality is to use a specific DNN classifier, which rests its decisions on similar features, such as the end-of-query detector [55], i.e., a fixed-length acoustic embedding of the utterance computed by an LSTM, the ASR decoder related features, e.g., the entropy of the forward probability distribution (large entropy indicating nondevice-directed speech), and features related to the 1-best

Digital home assistants have a completely hands-free voice-controlled interface.

word/character sequence hypothesis. Those features are combined and input to a dedicated second-turn, device-directed speech DNN detector.

An additional source of information that detects device-directed speech is derived from the speaker characteristic, because the second turn is spoken by the same speaker as the first. Actually, all speech following the wake-up keyword and spoken by the same speaker is considered to be device directed. Thus, a speaker-embedding vector can be computed from the detected keyword speech. This embedding can be used to make an acoustic beamformer speaker dependent [56], and to improve the end-of-query and second-turn, device-directed speech detection [57]. The encoder for mapping the keyword speech to an embedding is learned jointly with the classifier-detecting device-directedness. Thus, the classifier learns in a data-driven way what speaker and speech characteristics are relevant for detecting device-directedness.

Speaker identification

When digital home assistants are used by multiple members of a household, it is necessary to understand what the user is asking for and who the user is. The latter is important to correctly answer queries such as “When is my next appointment?” To do so, the system must perform utterance-by-utterance speaker identification. Speaker identification algorithms can be text-dependent, typically based on the wake-up keyword, [58], [59] or text independent [60], [61], and run locally on the device or on the server. An enrollment process is usually necessary so that the assistant can associate speech with a user profile. Enrollment can be implemented by explicitly asking a user to provide an identity and a few example phrases. An alternative approach is for the assistant to identify speakers based on analyzing past utterances, and, next time, when hearing a known speaker ask for providing an identity.

Case study

To illustrate the impact of front-end multichannel signal processing on ASR, the engineering team at Apple evaluated the performance of the far-field Siri speech processing system on a large speech test set recorded on HomePod in several acoustic conditions [6], such as

- music and podcast playback at different levels
- continuous background noise, including babble and rain noise
- directional noises generated by household appliances such as a vacuum cleaner, hairdryer, and microwave
- interference from external competing sources of speech.

In these recordings, the locations of HomePod and the test subjects were

varied to cover different use cases, e.g., in living room or kitchen environments where HomePod was placed against the wall or in the middle of the room.

The performance of Siri online multichannel signal processing was investigated in a real-world setup, where the trigger detection and subsequent voice command recognition jointly affect the user experience. Therefore, two objective Siri performance metrics, i.e., the FRRs and WERs, are reported.

Figure 5 shows the FRRs. The triggering threshold is the same in all conditions to keep the false alarm rates to a minimum. It can be observed that mask-based NR is suitable in most acoustic conditions except for the multitalker scenario, which is well handled by the stream selection system. For instance, in the competing talker case, the absolute FRR improvement of the multistream system is 29.0% when compared to that of mask-based NR, which has no source separation capability, and 30.3% when compared to the output of the baseline digital signal processing (DSP) system (which includes echo cancellation and dereverberation). The gap between mask-based NR and the multistream system becomes smaller in other acoustic conditions. Overall, there is a clear trend of healthy voice trigger detection improvement when mask-based NR and source separation techniques (stream selection) are used.

Figure 6 shows the WERs achieved by combining multichannel signal processing based on DL with the speech recognizer trained offline using internally collected live data from HomePod to augment an existing training set, which was found to improve ASR performance [6]. More details on data combination strategies to train AMs can be found in [2] and [3]. The blue portion of the bar represents the error rate of the triggered utterances, and the green portion represents the error rate due to falsely rejected utterances (missed utterances). Because triggered utterances can be different using one processing algorithm or another in different acoustic conditions, the WER numbers are directly influenced by the trigger performance. Different numbers of words are used for evaluation in the blue portion of the bars because the corresponding number of false rejections are significantly different for each

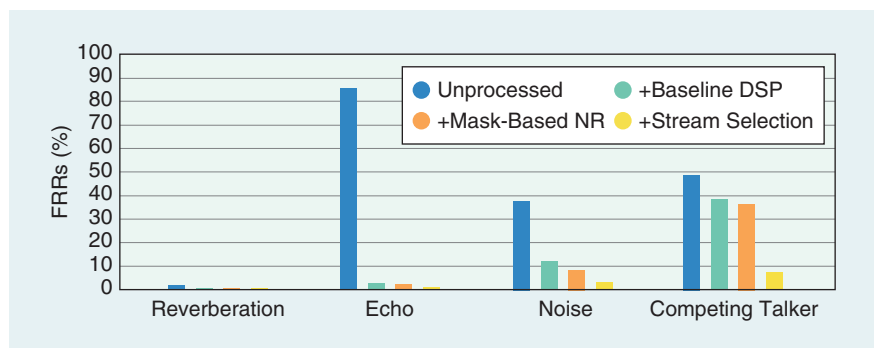


FIGURE 5. The FRRs of a “Hey Siri” detector in the following acoustic conditions: reverberation, echo, noise, and a competing talker. +Baseline DSP refers to the baseline DSP. +Mask-Based NR refers to the baseline DSP and mask-based NR. +Stream Selection refers to the baseline DSP, mask-based NR, and stream selection [6].

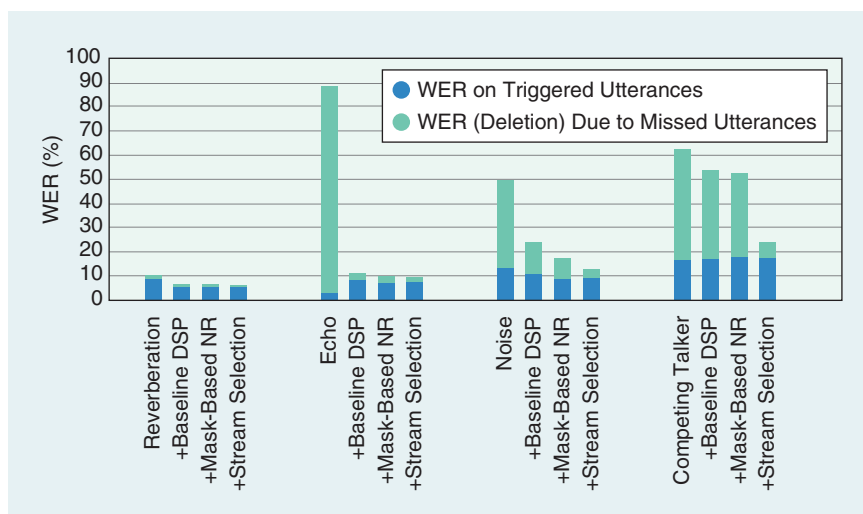


FIGURE 6. WERs in the following acoustic conditions: reverberation, an echo, noise, and a competing talker. +Baseline DSP refers to the baseline DSP, which includes echo cancellation and dereverberation. +Mask-based NR refers to the baseline DSP and mask-based NR +Stream Selection refers to the baseline DSP, mask-based NR, and stream selection [6].

case. It is obvious that the optimal and incremental integration of different speech processing technologies substantially improves the overall WER across conditions [6]. More specifically, the WER relative improvements are roughly 40, 90, 74, and 61%, respectively, in the four investigated acoustic conditions of reverberant speech only, playback, loud background noise, and competing talker [6].

Summary and outlook

This article provided an overview of the speech processing challenges and solutions of digital home assistants. While DL is the method of choice for overcoming many of these challenges, it is apparent that there is more to it than simply training a deep neural black-box classifier on sufficiently large data sets. A clever interplay of signal processing and DL needed to be developed to realize reliable, far-field, fully hands-free spoken interaction. The great success of this new class of products comes with new challenges, such as how to extend the range of applications and supported languages in an economically sensible way? Because of their conceptual simplicity, end-to-end ASR architectures appear to be one way to cope with those new challenges. However, more research is needed until those new concepts have proven effective for handling the unique and demanding challenges of smart loudspeakers. To see what is already possible today, please view an IEEE Signal Processing Society promotional video on YouTube [62], which illustrates that smart loudspeakers showcase signal processing at its best.

Authors

Reinhold Haeb-Umbach (haeb@nt.uni-paderborn.de) received his Dipl.-Ing. and Dr.-Ing. degrees from Rheinisch-Westfälische Technische Hochschule Aachen, Germany, in 1983 and 1988, respectively. He has a background in speech research in both industrial and academic research environ-

ments. Since 2001, he has been a professor of communications engineering at Paderborn University, Germany. His research interests are in the fields of statistical signal processing and pattern recognition, with applications to speech enhancement, acoustic beamforming and source separation as well as automatic speech recognition and unsupervised learning from speech and audio. He has coauthored more than 200 scientific publications including *Robust Automatic Speech Recognition—A Bridge to Practical Applications* (Academic Press, 2015). He is a fellow of the International Speech Communication Association.

Shinji Watanabe (shinjiw@ieee.org) received his B.S., M.S., and Ph.D. (Dr. Eng.) degrees from Waseda University, Tokyo, Japan, in 1999,

2001, and 2006, respectively. He is an associate research professor at Johns Hopkins University, Baltimore, Maryland. He was a research scientist at the Nippon Telegraph and Telephone Communication Science Laboratories, Kyoto, Japan, from 2001 to 2011; a visiting scholar at the Georgia Institute of Technology, Atlanta, in 2009; and a senior principal research scientist at Mitsubishi Electric Research Laboratories, Cambridge, Massachusetts, from 2012 to 2017. His research interests include automatic speech recognition, speech enhancement, spoken language understanding, and machine learning for speech and language processing. He has published more than 150 papers and has received several awards, including the Best Paper Award from the Institute of Electronics, Information and Communication Engineers in 2003.

Tomohiro Nakatani (tnak@ieee.org) received his B.E., M.E., and Ph.D. degrees from Kyoto University, Japan, in 1989, 1991, and 2002, respectively. He is a senior distinguished researcher at the Nippon Telegraph and Telephone Communication Science Laboratories, Kyoto, Japan. He was a visiting scholar at the Georgia Institute of Technology, Atlanta, in 2005 and a visiting assistant professor at Nagoya University, Japan, from 2008 to 2017. He is a member of the IEEE Signal Processing Society Speech and Language Processing Technical Committee. His research interests are in audio signal processing technologies for intelligent human-machine interfaces, including dereverberation, denoising, source separation, and robust automatic speech recognition.

Michiel Bacchiani (michi@b@google.com) received his Ingenieur (ir.) degree from Technical University of Eindhoven, The Netherlands, and his Ph.D degree from Boston University, Massachusetts. He has been a speech researcher with Google since 2005. Currently, he manages a research group at Google Tokyo, which is focused on the

joint modeling of speech and natural language understanding. Previously, he managed the acoustic modeling team responsible for developing novel algorithms and training infrastructure for all speech recognition applications that back Google services. Prior to joining Google, he worked as a member of the technical staff at IBM Research, as a technical staff member at AT&T Labs Research, and as a research associate at Advanced Telecommunications Research in Kyoto, Japan.

Björn Hoffmeister (bhoffmeister@apple.com) received his M.S. degree from Lübeck University, Germany, and his Ph.D. degree from RWTH Aachen University, Germany. He was a senior science manager at Amazon, leading Alexa Speech, the automatic speech recognition R&D group. He joined Amazon in 2011 as a founding member of Alexa research and developed the wake-word detection solution. Following the launch of Echo in 2014, he managed and grew the speech R&D group, which supports speech for all Amazon Alexa devices across all languages. In 2015, he led R&D efforts for the Alexa Skills Kit project. In 2016, he helped to launch Amazon Web Services' Lex service, which is based on Alexa speech and skills technology. In July 2019, he joined the Siri Team at Apple.

Michael L. Seltzer (mikeseltzer@fb.com) received his Sc.B. degree with honors from Brown University, Providence, Rhode Island, in 1996 and his M.S. and Ph.D. degrees from Carnegie Mellon University, Pittsburgh, Pennsylvania in 2000 and 2003, respectively. Currently, he is a research scientist in the Applied Machine Learning Division of Facebook. From 1998 to 2003, he was a member of the Robust Speech Recognition group at Carnegie Mellon University. From 2003 to 2017, he was a member of the Speech and Dialog Research group at Microsoft Research. In 2006, he received the IEEE Signal Processing Society Best Young Author Award for his work optimizing microphone array processing for speech recognition. His research interests include speech recognition in adverse environments, acoustic modeling and adaptation, neural networks, microphone arrays, and machine learning for speech and audio applications.

Heiga Zen (heigazen@google.com) received his A.E. degree from Suzuka National College of Technology, Japan, in 1999 and his Ph.D. degree from the Nagoya Institute of Technology, Japan, in 2006. Currently, he is a senior staff research scientist at Google. He was an intern/co-op researcher at the IBM T.J. Watson Research Center, Yorktown Heights, New York, from 2004 to 2005, and a research engineer in the Cambridge Research Laboratory at Toshiba Research Europe Ltd., United Kingdom, from 2008 to 2011. While at Google, he was with the Speech Team from July 2011 to July 2018 and joined the Brain Team in August 2018.

Mehrez Souden (msouden@apple.com) received his Ph.D. and M.Sc. degrees from the Institut National de la Recherche Scientifique, University of Québec, Montréal, Canada, in 2010 and 2006, respectively. Currently, he is a senior audio and speech processing engineer with Interactive Media Group, Apple Inc. He was with the Nippon Telegraph

and Telephone Communication Science Laboratories, Kyoto, Japan, from 2010 to 2012; the School of Electrical and Computer Engineering, the Georgia Institute of Technology, Atlanta, from 2013 to 2014; and the Intel Corporation from 2014 to 2015. In 2016, he joined Apple Inc. to work on signal processing and machine learning with an emphasis on acoustics and speech. He has published more than 50 papers. He received the Alexander Graham Bell Canada Graduate Scholarship and a postdoctoral fellowship, both from the National Sciences and Engineering Research Council, in 2008 and 2013, respectively.

References

- [1] E. Vincent, S. Watanabe, A. A. Nugraha, J. Barker, and R. Marxer, "An analysis of environment, microphone and data simulation mismatches in robust speech recognition," *J. Comput. Speech Language*, vol. 46, pp. 535–557, Nov. 2017. doi: 10.1016/j.csl.2016.11.005.
- [2] J. Barker, S. Watanabe, E. Vincent, and J. Trmal, "The fifth CHiME speech separation and recognition challenge: Dataset, task and baselines," in *Proc. INTERSPEECH*, 2018, pp. 1561–1565.
- [3] K. Kinoshita, M. Delcroix, S. Gannot, E. Habets, R. Haeb-Umbach, W. Kellermann, V. Leutnant, R. Maas et al., "A summary of the REVERB challenge: State-of-the-art and remaining challenges in reverberant speech processing research," in *Proc. EURASIP Journal on Advances in Signal Processing*, 2016. doi: 10.1186/s13634-016-0306-6.
- [4] M. Harper, "The automatic speech recognition in reverberant environments (ASpIRE) challenge," in *Proc. IEEE Automatic Speech Recognition and Understanding (ASRU) Workshop*, 2015, pp. 547–554.
- [5] T. Yoshioka, A. Sehr, M. Delcroix, K. Kinoshita, R. Maas, T. Nakatani, and W. Kellermann, "Making machines understand us in reverberant rooms: Robustness against reverberation for automatic speech recognition," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 114–126, Nov 2012.
- [6] Audio Software Engineering and Siri Speech Team, "Optimizing Siri on HomePod in far-field settings," *Mach. Learn. J.*, vol. 1, no. 12, 2018. [Online]. Available: <https://machinelearning.apple.com/2018/12/03/optimizing-siri-on-homepod-in-far-field-settings.html>
- [7] J. Benesty, T. Gänslér, D. Morgan, M. Sondhi, and S. Gay, *Advances in Network and Acoustic Echo Cancellation*. New York: Springer-Verlag, 2001.
- [8] B. Schwartz, S. Gannot, and E. A. P. Habets, "Online speech dereverberation using Kalman filter and EM algorithm," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 23, no. 2, pp. 394–406, 2015.
- [9] T. Nakatani, T. Yoshioka, K. Kinoshita, M. Miyoshi, and B.-H. Juang, "Blind speech dereverberation with multi-channel linear prediction based on short time Fourier transform representation," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2008, pp. 85–88.
- [10] T. Yoshioka and T. Nakatani, "Generalization of multi-channel linear prediction methods for blind MIMO impulse response shortening," *IEEE Trans. Audio, Speech, and Language Process.*, vol. 20, no. 10, pp. 2707–2720, 2012.
- [11] L. Drude, C. Boeddeker, J. Heymann, K. Kinoshita, M. Delcroix, T. Nakatani, and R. Haeb-Umbach, "Integrating neural network based beamforming and weighted prediction error dereverberation," in *Proc. INTERSPEECH*, 2018, pp. 3043–3047.
- [12] T. Yoshioka, H. Tachibana, T. Nakatani, and M. Miyoshi, "Adaptive dereverberation of speech signals with speaker-position change detection," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2009, pp. 3733–3736.
- [13] J. Caroselli, I. Shafran, A. Narayanan, and R. Rose, "Adaptive multichannel dereverberation for automatic speech recognition," in *Proc. INTERSPEECH*, 2017, pp. 1701–1791.
- [14] CHiME Challenge, "The 5th CHiME speech separation and recognition challenge: Results." Accessed on: Dec. 7, 2018. [Online]. Available: http://spandh.dcs.shef.ac.uk/chime_challenge/results.html
- [15] C. Boeddeker, H. Erdogan, T. Yoshioka, and R. Haeb-Umbach, "Exploring practical aspects of neural mask-based beamforming for far-field speech recognition," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 6697–6701.
- [16] J. Heymann, M. Bacchiani, and T. Sainath, "Performance of mask based statistical beamforming in a smart home scenario," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 6722–6726.
- [17] N. Ito, S. Araki, and T. Nakatani, "Complex angular central Gaussian mixture model for directional statistics in mask-based microphone array signal processing," in *Proc. European Signal Processing Conf. (EUSIPCO)*, 2016, pp. 1153–1157.

- [18] D. H. Tran Vu and R. Haeb-Umbach, "Blind speech separation employing directional statistics in an expectation maximization framework," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2010, pp. 241–244.
- [19] N. Q. Duong, E. Vincent, and R. Gribonval, "Under-determined reverberant audio source separation using a full-rank spatial covariance model," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 18, no. 7, pp. 1830–1840, 2010.
- [20] O. Yilmaz and S. Rickard, "Blind separation of speech mixtures via time-frequency masking," *IEEE Trans. Signal Process.*, vol. 52, no. 7, pp. 1830–1847, 2004.
- [21] S. Gannot, E. Vincent, S. Markovich-Golan, and A. Ozerov, "A consolidated perspective on multimicrophone speech enhancement and source separation," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 25, no. 4, pp. 692–730, 2017.
- [22] J. Heymann, L. Drude, and R. Haeb-Umbach, "Neural network based spectral mask estimation for acoustic beamforming," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 196–200.
- [23] H. Erdogan, J. R. Hershey, S. Watanabe, M. I. Mandel, and J. Le Roux, "Improved MVDR beamforming using single-channel mask prediction networks," in *Proc. INTERSPEECH*, 2016, pp. 1981–1985.
- [24] J. Heymann, L. Drude, and R. Haeb-Umbach, "A generic neural acoustic beamforming architecture for robust multi-channel speech processing," *J. Comput. Speech Language*, vol. 46, no. C, pp. 374–385, 2017.
- [25] Y. Liu, A. Ganguly, K. Kamath, and T. Kristjansson, "Neural network based time-frequency masking and steering vector estimation for two-channel MVDR beamforming," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 6717–6721.
- [26] M. Pedersen, J. Larsen, U. Kjems, and L. Parra, "A survey of convolutive blind source separation methods," in *Springer Handbook Speech Processing and Speech Communication*, Jacob Benesty, Yiteng Huang, Mohan Sondhi, Eds. New York: Springer, Nov. 2007, pp. 114–126. [Online]. Available: http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/4924/pdf/imm4924.pdf
- [27] J. Hershey, Z. Chen, J. L. Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 31–35.
- [28] D. Yu, M. Kolbaek, Z.-H. Tan, and J. Jensen, "Permutation invariant training of deep models for speaker-independent multi-talker speech separation," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 241–245.
- [29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition." 2015. [Online]. Available: <http://arxiv.org/abs/1512.03385>
- [30] J. B. Allen and D. Berkley, "Image method for efficiently simulating small-room acoustics," *J. Acoust. Soc. Amer.*, vol. 65, no. 4, pp. 943–950, 1979.
- [31] E. Hadad, F. Heese, P. Vary, and S. Gannot, "Multichannel audio database in various acoustic environments," in *Proc. Int. Workshop Acoustic Signal Enhancement (IWAENC)*, 2014, pp. 313–317.
- [32] C. Kim, A. Misra, K. Chin, T. Hughes, A. Narayanan, T. Sainath, and M. Bacchiani, "Generation of large-scale simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in Google Home," in *Proc. INTERSPEECH*, 2017, pp. 379–383.
- [33] M. L. Seltzer, B. Raj, R. M. Stern, "Likelihood-maximizing beamforming for robust hands-free speech recognition," *IEEE Speech Audio Process.*, vol. 12, no. 5, pp. 489–498, 2004.
- [34] T. N. Sainath, R. J. Weiss, K. W. Wilson, B. Li, A. Narayanan, E. Variiani, M. Bacchiani, I. Shafraan et al., "Multichannel signal processing with deep neural networks for automatic speech recognition," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 25, no. 5, pp. 965–979, 2017.
- [35] X. Xiao, S. Watanabe, H. Erdogan, L. Lu, J. Hershey, M. L. Seltzer, G. Chen, Y. Zhang et al., "Deep beamforming networks for multi-channel speech recognition," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 5745–5749.
- [36] J. Heymann, L. Drude, C. Boeddeker, P. Hanebrink, and R. Haeb-Umbach, "BEAMNET: End-to-end training of a beamformer-supported multi-channel ASR system," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 5325–5329.
- [37] C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss et al., "State-of-the-art speech recognition with sequence-to-sequence models." 2017. [Online]. Available: <http://arxiv.org/abs/1712.01769>
- [38] T. Ochiai, S. Watanabe, T. Hori, and J. R. Hershey, "Multichannel end-to-end speech recognition," in *Proc. Int. Conf. Machine Learning (ICML)*, 2017.
- [39] B. Li, T. N. Sainath, A. Narayanan, J. Caroselli, M. Bacchiani, A. Misra, I. Shafraan, H. Sak et al., "Acoustic modeling for Google Home," in *Proc. INTERSPEECH*, 2017, pp. 399–403.
- [40] P. S. Aleksic, M. Ghodsi, A. H. Michaely, C. Allauzen, K. B. Hall, B. Roark, D. Rybach, and P. J. Moreno, "Bringing contextual information to Google speech recognition," in *Proc. INTERSPEECH*, 2015, pp. 468–472.
- [41] A. Raju, B. Hedayatnia, L. Liu, A. Gandhe, C. Khatri, A. Metallinou, A. Venkatesh, and A. Rastrow, "Contextual language model adaptation for conversational agents," in *Proc. INTERSPEECH*, 2018, pp. 3333–3337.
- [42] H. Lin, J. Huang, F. Beaufays, B. Strophe, and Y. Sung, "Recognition of multilingual speech in mobile applications," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 4881–4884.
- [43] T. Capes, P. Coles, A. Conkie, L. Golipour, A. Hadjitarkhani, Q. Hu, N. Huddleston, M. Hunt et al., "Siri on-device deep learning-guided unit selection text-to-speech system," in *Proc. INTERSPEECH*, 2017, pp. 4011–4015.
- [44] A. van den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. van den Driessche, E. Lockhart et al., "Parallel WaveNet: Fast high-fidelity speech synthesis," in *Proc. Int. Conf. Machine Learning (ICML)*, 2018, pp. 3918–3926.
- [45] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior et al., "Wavenet: A generative model for raw audio." 2016. [Online]. Available: <http://arxiv.org/abs/1609.03499>
- [46] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 4087–4091.
- [47] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Proc. INTERSPEECH*, 2015, pp. 1478–1482.
- [48] K. Kumatani, S. Panchapagesan, M. Wu, M. Kim, N. Strom, G. Tiwari, and A. Mandai, "Direct modeling of raw audio with DNNs for wake word detection," in *Proc. IEEE Automatic Speech Recognition and Understanding (ASRU) Workshop*, 2017, pp. 252–257.
- [49] S. Fernández, A. Graves, and J. Schmidhuber, "An application of recurrent neural networks to discriminative keyword spotting," in *Proc. 17th Int. Conf. Artificial Neural Networks*, 2007, pp. 220–229.
- [50] M. Wu, S. Panchapagesan, M. Sun, J. Gu, R. Thomas, S. N. P. Vitaladevuni, B. Hoffmeister, and A. Mandal, "Monophone-based background modeling for two-stage on-device wake word detection," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5494–5498.
- [51] Siri Team, "Hey Siri: An on-device DNN-powered voice trigger for Apple's personal assistant," *Mach. Learn. J.*, vol. 1, no. 6, 2017. [Online]. Available: <https://machinelearning.apple.com/2017/10/01/hey-siri.html>
- [52] M. Shannon, G. Simko, S.-y. Chang, and C. Parada, "Improved end-of-query detection for streaming speech recognition," in *Proc. INTERSPEECH*, 2017, pp. 1909–1913.
- [53] B. Liu, B. Hoffmeister, and A. Rastrow, "Accurate endpointing with expected pause duration," in *Proc. INTERSPEECH*, 2015, pp. 2912–2916.
- [54] R. Maas, A. Rastrow, C. Ma, G. Lan, K. Goehner, G. Tiwari, S. Joseph, and B. Hoffmeister, "Combining acoustic embeddings and decoding features for end-of-utterance detection in real-time far-field speech recognition systems," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5544–5548.
- [55] S. Mallidi, R. Maas, K. Goehner, R. A. S. Matsoukas, and B. Hoffmeister, "Device-directed utterance detection," in *Proc. INTERSPEECH*, 2018, pp. 1225–1228.
- [56] J. Wang, J. Chen, D. Su, L. Chen, M. Yu, Y. Qian, and D. Yu, "Deep extractor network for target speaker recovery from single channel speech mixtures," in *Proc. INTERSPEECH*, 2018, pp. 307–311.
- [57] R. Maas, S. H. K. Parthasarathi, B. King, R. Huang, and B. Hoffmeister, "Anchored speech detection," in *Proc. INTERSPEECH*, 2016, pp. 2963–2967.
- [58] E. Variiani, X. Lei, E. McDermott, I. Lopez-Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 4052–4056.
- [59] S.-X. Zhang, Z. Chen, Y. Zhao, J. Li, and Y. Gong, "End-to-end attention based text-dependent speaker verification," in *Proc. IEEE Spoken Language Technology (SLT) Workshop*, 2016, pp. 171–178.
- [60] C. Zhang and K. Koishida, "End-to-end text-independent speaker verification with triplet loss on short utterances," in *Proc. INTERSPEECH*, 2017, pp. 1487–1491.
- [61] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5329–5333.
- [62] YouTube, "Signal processing in home assistants." Accessed on: Dec. 7, 2018. [Online]. Available: <https://www.youtube.com/watch?v=LJ54btWttdo>

Muhammad Salman Khan, Millaray Curilem, Fernando Huenupán,
Muhammad Farhan Khan, and Néstor Becerra Yoma

A Signal Processing Perspective of Monitoring Active Volcanoes

A volcano is a rupture in Earth's crust that unleashes lava, volcanic ash, and gases from a cavity beneath the surface. Monitoring volcanoes is crucial for getting an insight into the dynamics of the different events taking place, modeling their behavior, and establishing mechanisms to efficiently deal with eruptions. The seismic signals coming from volcanoes are a great source of information about what happens inside their structure and thus are helpful in forecasting their activity. Their study is important to ensure the safety of people who live in the vicinity of different volcanoes around the world. This is why volcanic-activity monitoring is being supported by applying modern signal processing and machine-learning methods.

Seismic events are observed through different instruments that translate the movements of Earth into measurable signals. Signal processing plays a critical role in analyzing these signals and relating them to specific events occurring inside the volcano. Signal processing is a vital component of seismic event observation that enables automatic volcano activity detection and classification. To better understand and visualize the different signal processing stages, we present our experience in exploring the active volcanoes of Chile. We utilize real data as a case study in this article to highlight the key signal processing steps in active volcano monitoring.

Chile has a high concentration of active volcanoes due to its location along the Peru–Chile Trench, marking the subduction of the Nazca Plate under the South American Plate. A list of the five most active volcanoes is given in Table 1. The Llaima volcano, shown in Figure 1, is located in the Araucanía region of Chile, and it is considered one of the most active volcanoes in South America and the second most dangerous in Chile after Villarrica. The Southern Andean Volcano Observatory (OVDAS), a program of the Chilean National Geology and Mining Service, monitors the volcanic activity in Chile.

Main elements in active volcano monitoring

Figure 2 depicts the typical stages involved in active volcano monitoring. Stations for monitoring the volcano-seismic activities are responsible for recording different types of signals. These signals are sent to an observatory through a data-transmission network. Each station transmits its data and is gen-

erally made up of the elements described as follows:

- **Seismic sensors:** Seismic sensors are devices meant to measure seismic vibrations that travel from inside the volcano. They must be buried in the ground at a depth of approximately 60 cm to 1 m on a solid surface (i.e., a rock) so that they capture the signal in the best conditions and with minimum noise. The sensors must have adequate polyethylene insulation for them to capture minimum noise from the environment.
- **Digitizer:** A digitizer is responsible for transforming the voltage sent by the sensor to a form that can be read and transmitted by the radios. The gain of the digitizer modifies the sensitivity and limits the upper frequency band of the sensors. The sampling frequency is typically 50–100 Hz.
- **Communication system:** The communication system consists of a radio and antenna that send signals from the digitizer to another radio

Table 1. The top five most active volcanoes of Chile.

| Ranking | Name | Coordinates (Latitude, Longitude) | Most Recent Eruption (Year, Common Era) |
|---------|------------|-----------------------------------|-----------------------------------------|
| 1 | Villarrica | 39.42 °S, 71.93 °W | 2018 |
| 2 | Llaima | 38.692 °S, 71.729 °W | 2009 |
| 3 | Calbuco | 41.33 °S, 72.618 °W | 2015 |
| 4 | Chaitén | 42.833 °S, 72.646 °W | 2011 |
| 5 | Láscar | 23.37 °S, 67.73 °W | 2017 |

(Source: Data are from the Global Volcanism Program, Smithsonian Institution.)

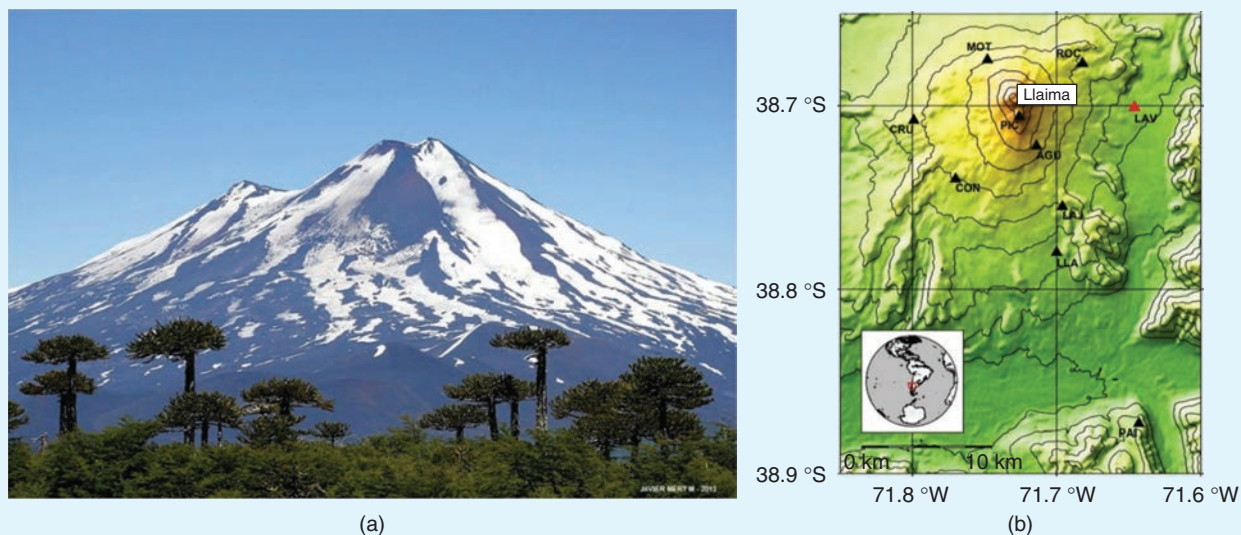


FIGURE 1. (a) The Llama volcano and (b) its seismic stations. Llama last erupted in 2009. (Source: <http://www.sernageomin.cl>; used with permission.)

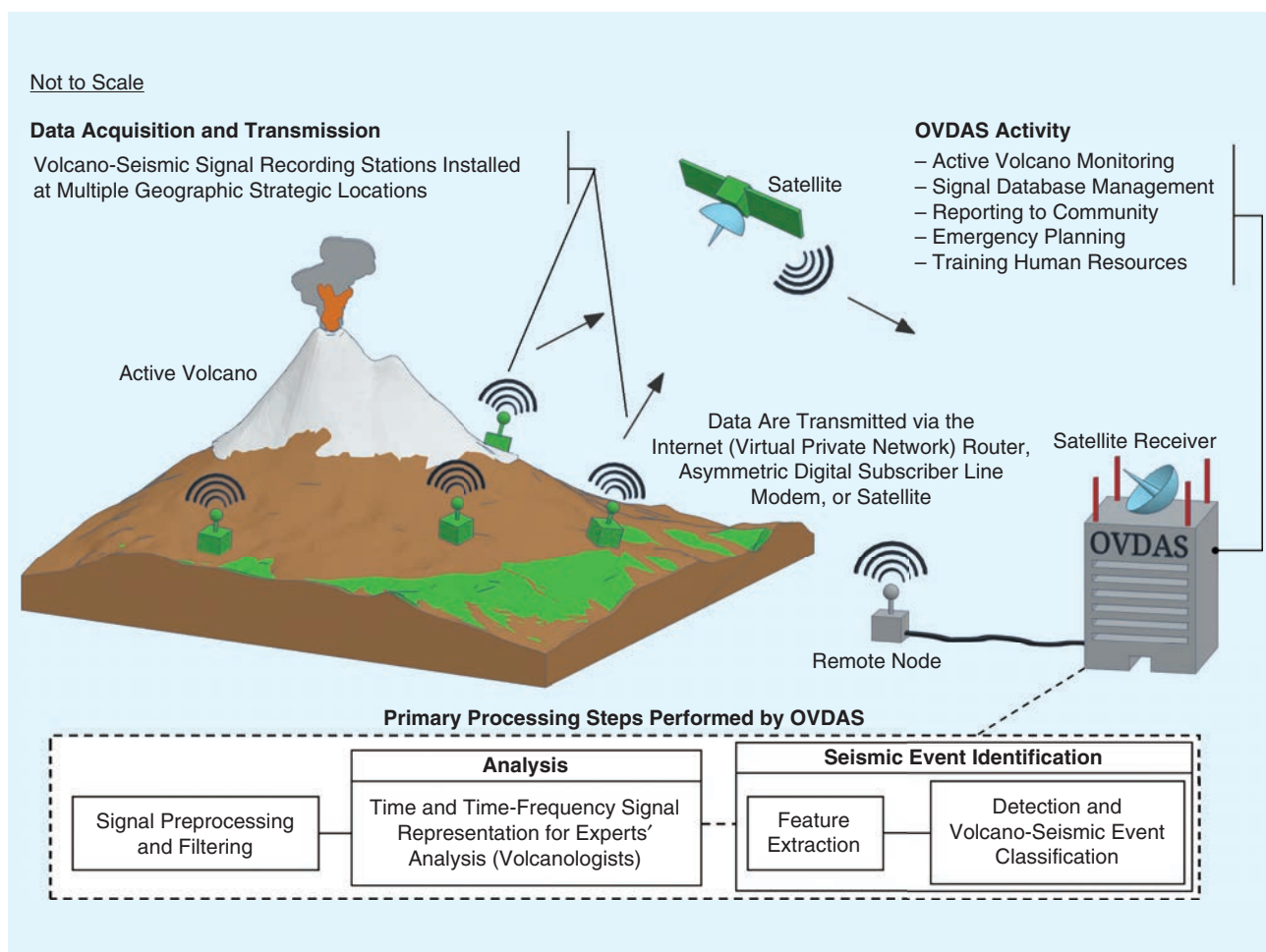


FIGURE 2. A depiction of seismic signal measurement, transmission, reception, and analysis for event detection and recognition.

and antenna. These data are sent by telemetry. There are some stations that, because of location problems (hills that interrupt the path of the data), cannot send their data directly to a central computing node. In those cases, repeaters are installed.

- **Computing node:** Remote communications stations are responsible for receiving information from the monitoring stations and forwarding it, either through a satellite link to the hub (a central station where the data transmitted from the nodes will be received and sent to the data center—for instance, OVDAS—for further processing) or through the Internet (this node can be any public place that has access to the Internet—for example, municipalities, schools, and police stations).
- **Power supply:** Because the stations are in the vicinity of the volcano, they are far from the high-, medium-, and low-power lines. Therefore, the devices cannot be powered through the conventional power distribution network. That is why solar panels are placed in firm iron structures. These solar panels are connected to a bank of deep-cycle batteries through a charge controller, where the digitizer is also connected.

Volcano-seismic activity

The internal environment of a volcano has an activity characterized by fluid transportation, releasing energy that generates different physical processes, such as explosions (EXs), rock fracturing, degasification, magmatic intrusion, eruptions, pressurization, and depressurization [49]. Some of these physical processes are referred to as *volcanic-seismic events* and can be identified by analyzing discrete oscillations present in signals monitored from the seismic stations. A portion of the signal is considered an event when it presents typical characteristics in magnitude, dynamics of the time and frequency waveform, and duration.

Figure 3 shows plots in time and spectrogram domains of several types of volcano events: long-period (LP)

events, volcano-tectonic (VT) earthquakes, tremors (TRs), tornillos (TOs), EXs, distal VT (dVT) earthquakes, hybrid events (HBs), avalanches (AVs), and ice quakes (IQs). Each event was manually segmented and classified by an OVDAS analyst from the continuous signal obtained from seismographs within the volcano monitoring network. It is evident from Figure 3(a)–(i) that every volcano event has its unique time duration, spectrogram shape, and dominant frequency. Along with other characteristics, these help in discriminating among and classifying different events. A brief description of some of the important events is provided as follows:

- **LP events:** The origin of LP events has been associated with movements of fluids and resonances generated by pressure increases within fractures saturated with fluids. However, new studies have associated their origin to the rupture of magma in the walls of the canal. Their duration does not usually exceed 1 min. The main characteristic of an LP event is that it concentrates its energy at low frequencies, usually below 3 Hz.
- **VT events:** VT events come from the rupture of a rock due to volcanic processes. The temporary signals of the VT have large amplitudes with exponential decay, and they last for a shorter duration than with the LP events. In the frequency domain, their energy is concentrated in the range between 6 and 8 Hz. They are difficult to detect in the presence of other events or noise. In the Llaïma volcano, they are important because they occur mainly during eruptions. Due to this fact, VT events occur less frequently than LP events [1].
- **TR events:** TR events are linked to the transit and dynamics of fluids inside the volcanic building. They have longer durations (several minutes to hours). In the frequency domain, they have a stable behavior and are centered in narrow frequency bands (1–5 Hz). The spectral content of the volcanic TR is very similar to that of the LP event, and this is explained by the proposed source mechanisms (i.e., resonant fluid) [2].

- **Tectonic (TC) events:** TC events are related to the movement of plates and deformations of Earth's crust, notwithstanding volcanic activity. In general, they are characterized by a wide frequency spectrum similar to that of VT events, but of longer duration.

Signal preprocessing and representation

Once the signals have been received in the observatory, they are preprocessed/filtered for expert analysis. Generally, the following type of preprocessing is employed. The sampling frequency of the signals is typically normalized at 100 Hz. The amplitude of the signals is represented in micrometers per second. Usually, a Butterworth bandpass filter is utilized. The bandwidth is between 1 and 12 Hz. However, the cutoff frequencies and the filter's order vary according to the volcano.

Figure 4 depicts volcano-seismic signals observed at different seismic stations of the Llaïma volcano. This volcano has five stations: Laguna Verde, Motin, Condor, Aguila, and Llaite. The signals are shown in the time domain. Certain events detected and classified by volcanologists are represented in the time-frequency or spectrogram domain. For instance, the time-domain image titled “FREZ HHZ TC 99” indicates the continuous data visualization for one single day from the Fresco station at the *z* component, corresponding to the vertical axis. Therefore, “FREZ” indicates the station and the signal component visualized, and “HHZ TC 99” denotes the communication channel (i.e., where the data are saved). The selected stations are those that provide the best representations for each class and show the best availability over time, because some of them can be down due to technical problems, such as a low power supply. The images were filtered by analysts using multiple filters that were chosen based on their experience.

Automatic detection and classification

The improvement in accuracy of automatic volcano-seismic event detection

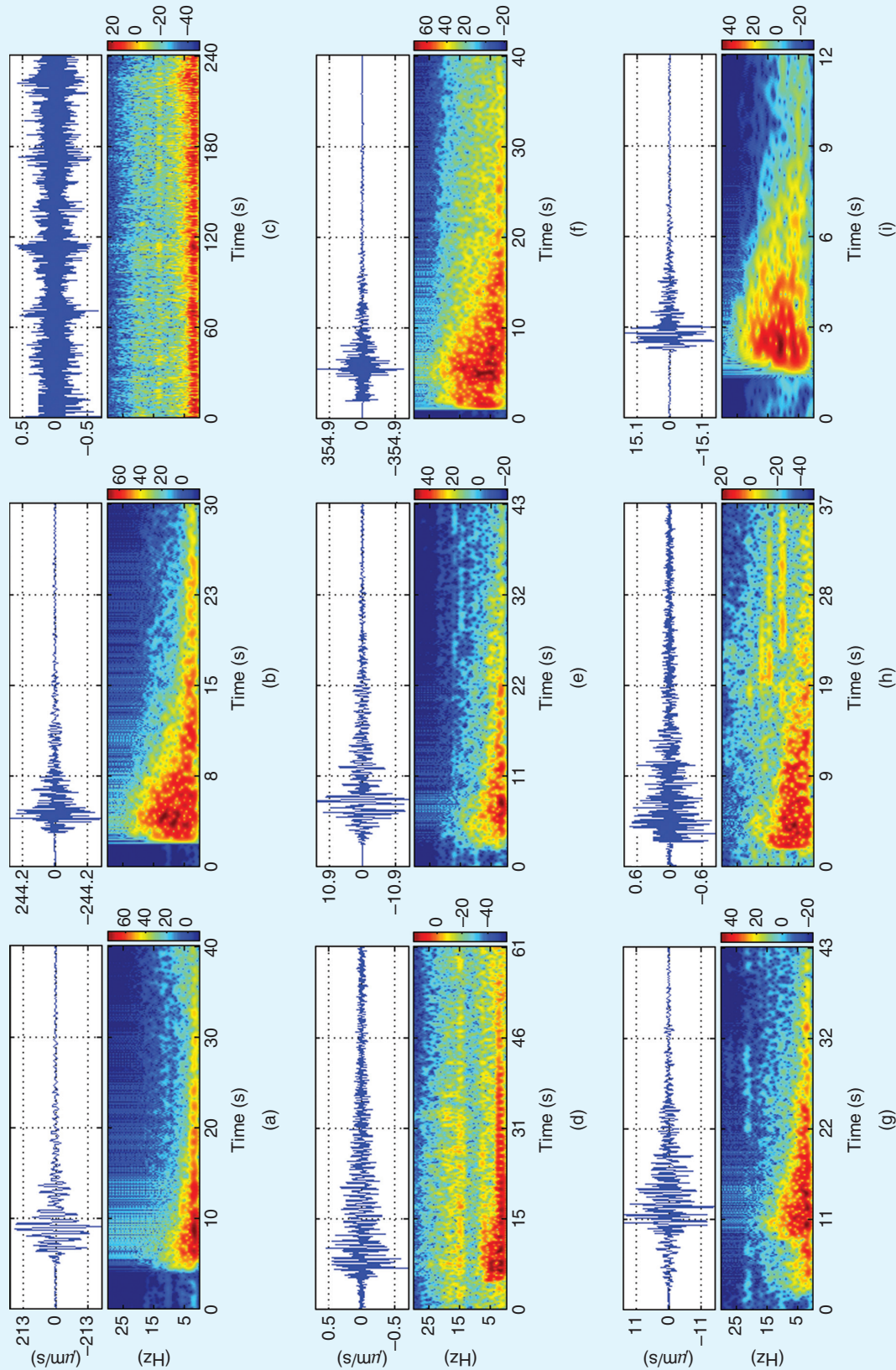


FIGURE 3. The signals shown in the time and spectrogram domains are related to the following events: (a) LP events, (b) VT earthquakes, (c) TRs, (d) TOs, (e) EXs, (f) dVT earthquakes, (g) HBs, (h) AVs, and (i) IQs. The signals were filtered using a 10th-order Butterworth bandpass filter between 0.8–20 Hz. The spectrograms were generated with a 1-s Hamming window (for 100 samples), a 98% overlap, and 1,024 frequency samples.

and classification can be achieved by using suitable descriptors (features) for the event signals. The goals are to get maximum information from the signals and to reduce the data dimensionality by converting the sampled signal into a sequence of parameter vectors with less redundant information.

Feature selection

The features extracted have to be appropriate for the classification problem so that they may retrieve enough information to divide the classification space into decision regions associated to the different classes (events). Additionally, the number of features must be reduced to limit the complexity of the classifier. This is why a feature selection process is needed to define a minimum set of features that maximizes the classification performance [3]. As a result of this process, some features may become unnecessary when combined with other parameters due to possible redundancies. Therefore, several feature selection methods are used to obtain a reduced set of features and to, at the same time, improve the performance of classification systems. In [3] and [57], different criteria are presented to select characteristics, highlighting those based on correlation that allow the identification of redundant features [58]. This process can be complex if large sets of features with similar behaviors are used.

Many techniques have been proposed in the literature to perform feature dimension reduction for volcano event classification [4], such as self-organizing maps [5], genetic algorithms [6], discriminant methods [7]–[9], and principal component analysis [10]. Also, in [11], several feature dimension reduction methods were compared. Wrapper methods, such as the discriminative feature selection technique [8] and genetic algorithms [6], have the advantage of preserving the feature space basis and taking advantage of the contribution of the features combination, making no assumptions on the original features' statistical distribution [11]. On the other hand, reduction techniques that transform the feature space, such as principal component analysis, independent component analysis, or Fisher

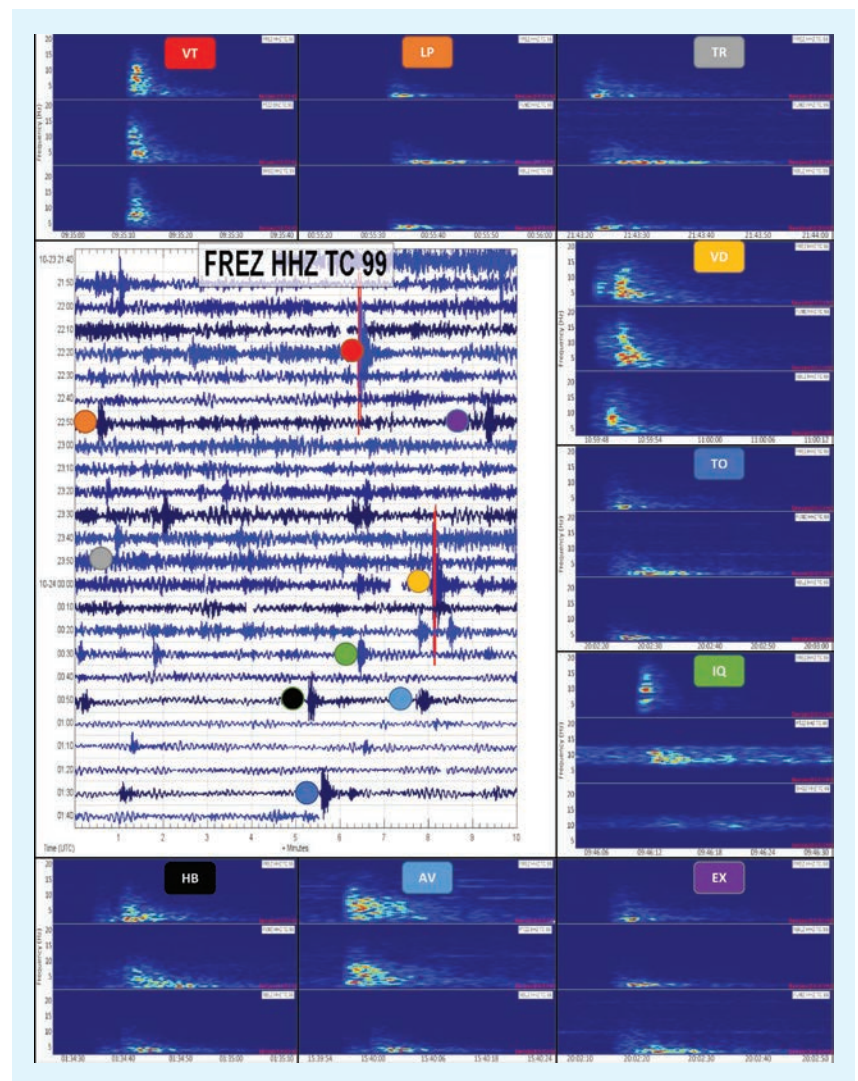


FIGURE 4. This image was generated with Swarm, a Java application designed to display and analyze seismic waveforms in real time. Spectrograms of different event classes recorded at three different stations are shown here.

discriminant analysis, could be more time-consuming for large databases. They need to perform preprocessing steps and to make some assumptions on the feature distributions. However, it is worth highlighting that, eventually, the best technique depends on the requirements of each task [11].

Feature extraction

In volcanic seismicity, the features are of geophysical and statistical nature, but most of them are extracted from the transform on the original data space [11]. Intensive research has been performed to define reliable features to accurately discriminate among volcano-seismic events [12]. Statistical dis-

tribution of the spectral characteristics of the signal and their evolution along time were used to define cepstral coefficients in [13]. In [14], the authors used amplitude statistics (mean, standard deviation, skewness, and kurtosis) and the power of the events to classify tectonic earthquakes. A linear predictive coding technique was also proposed to extract spectral features in combination with other parameterization methods to extract information about the waveform [15]. Cepstral coefficients [22] and linear prediction filter coefficients (LPCs) [23] are commonly used in feature extraction for speech technology applications. LPCs give compressed information about the shape of

the spectral envelope of a signal. They can be used to predict a signal sample through a linear combination of previous samples. Cepstral coefficients are defined as the discrete cosine transform coefficients obtained from the log magnitude over the fast Fourier transform (FFT) of the signal. This feature vector is related to the statistical distribution of the spectral characteristics of the signal [13].

In [16], spectral autocorrelation functions were employed by applying the FFT to represent the spectral content of the signals. An amplitude ratio between the bandpass filtered with unfiltered traces and the ratio between the maximum amplitude of the signal with the root mean square amplitude was applied. These amplitude ratios and the statistical moments help to distinguish between signal peak transients (such as VT events) from signals with long duration (e.g., rock falls) with similar frequency content.

Short-time FT was also applied to identify changes in the activity of Etna TR [17]. Wavelet techniques are well suited to the extraction of features from 2D imagery. This is why they have been used extensively to extract parameters from the time–frequency domain of seismic events [18]–[21]. As stated in [21], spatial and temporal correlations between adjacent scales often provide new physical insights on seismicity.

As spectrograms reflect the time dependence in the frequency content [24], they are widely applied in seismology [12]. In many works, features were extracted from the spectrograms to train the classifiers steps [25]–[27]. In [28], the authors used spectrograms of the seismic signals to understand the volcano internal conduit resonance. In [29], weighted eigenspectrograms were employed to remove the unwanted spectral content of the seismic signals, improving the signal-to-noise ratio. A new cluster evaluation criterion specific to the spectral space was proposed in [10] to analyze synthetic spectrograms and to discriminate their seismic origin. A comparative study was performed in [40] where the features selected to classify the events recorded at the Vil-

larrica volcano were used to classify events recorded at the Llaima volcano, which is a volcano with a similar structure and composition. The results showed that the performance of the classifiers decreased by 10%, indicating the need to investigate for new features. The study supports the observation that the feature extraction and selection processes are difficult to generalize. In [6], [39], [41]–[43], [46], [47], and [53]–[56], authors provide different insights on the feature extraction process with a variety of classification approaches.

Classification schemes

Several approaches have been used to address the automatic classification problem of volcanic seismic events. The lack of consensus regarding the techniques can be appreciated again at this stage. Support vector machines (SVMs) [30], self-organizing maps [31], [32], hidden Markov models (HMMs) [33], [34], Gaussian mixture models [8], multilayer perceptron (MLP) [23], [35], and deep neural networks (DNNs) [36] have been applied. A dictionary of seismic words was applied for the classification of volcano-seismic events of the Galeras volcano in Colombia [37]. In [36], a comparison showed that the DNN model achieved better results when compared with MLP, SVM, and random forest classifiers. In relation to the DNN model, according to the authors, the pretraining initialization was effective to support the training process. However, the results are difficult to be generalized because of the variability of the volcanoes, the events, and the features considered.

In [38], the authors performed an interesting analysis of the state of the art in machine-learning techniques for volcano-seismic signals. They divide the problem into detection and classification steps. In terms of detection, the authors point out that, although the results of the studies are promising, there is no established procedure to detect volcano-seismic events in continuous recordings when the number of signals is very high (e.g., during an eruption), when the signals are superimposed, or when the signals are slowly emerging. On the other hand, the methods proposed for the

automatic classification of volcano-seismic data are also promising, but they must be improved, as the studies consider just a limited number of signals, classes, and observation time intervals. This shows the need for increasing the research carried out in volcano observatories worldwide.

Most of works implemented for the Villarrica and Llaima volcanoes are based on the approach presented in [38]. First, the segmentation allows extraction of the event from the continuous signal. Then, the information retrieved by the feature extraction and selection stages is delivered to a classification structure based on SVMs. The emphasis of this approach is to find the best descriptors of the types of events analyzed, generally LP, TR, VT, and other types. So the main efforts have been to extract and select descriptors based on the literature and on other applications, such as speech [1], [6], [9], [40]. A particular study of the spectrograms was performed in [52] to evaluate different representations and their impact in the classification performance of five classes of events of the Llaima volcano. In [45], new features were created and extracted from the spectrograms [45], giving good classification results. To avoid the hard task of feature selection, a DNN approach was proposed in [44] to process the signals represented as spectrograms, providing promising results to discriminate among the different classes based on the shape of the time-frequency evolution of the events.

Another approach was performed in [47], where an automatic volcano event detection system based on the HMM with state and event duration models was proposed. The interesting point here is that different volcanic events have different durations. Therefore, the state and whole event durations learned from the training data were enforced on the corresponding state and event duration models within the HMM. This is an interesting alternative to the conventional machine-learning approaches that separates the detection and the classification of the events [23], [30], [35].

A preliminary study focused on the noise analysis of the seismic signals was performed in [48]. Traditional filters, such as Butterworth, spectral subtraction, and Wiener filters, were compared to the proposed modified spectral subtraction and modified Wiener filters. The aim of the study was to denoise the signals and to improve the localization and classification tasks. To support localization, a generalized cross-correlation method was employed to calculate the time difference of arrival using pairs of sensors, and a DNN technique was proposed to classify the three main volcano-seismic events occurring in the Llaima volcano. However, the presented methods still need to be tested on larger data sets.

Potential directions for future research

As we showed in this article, signal processing in combination with machine learning can enable highly accurate detection and classification systems. This provides a level of automation that can be of great help for human experts (volcanologists) in countries with a high density of monitored active volcanoes. Improving the accuracy of this technology is certainly an obvious direction for future research. The issue is how deep learning has led to impressive improvements in speech recognition, image recognition, scene understanding, natural language processing-based applications, and so on. However, deep learning requires a huge amount of training data to be useful. For example, some companies report speech recognition results by using large databases and very simple plain features with many thousands, if not millions, of hours of speech. Clearly, volcano databases are much smaller, and the number of events is limited. This does not imply that deep learning will not lead to improvements when compared to ordinary classification systems. Rather, it will strengthen the importance of engineered feature schemes. In this context, a proper or optimum representation of volcano events should help to improve the accuracy of machine-learning technology, including deep learning, with limited training data.

Acknowledgment

We thank the experts at the Observatorio Volcanológico de los Andes del Sur, Chile, for their support.

Authors

Muhammad Salman Khan (salman.khan@uetpeshawar.edu.pk) received his B.S. (honors), M.S., and Ph.D. degrees in electrical engineering from the University of Engineering and Technology, Peshawar, Pakistan, in 2007, The George Washington University, Washington, D.C., in 2010, and Loughborough University, United Kingdom, in 2013 respectively. He is an assistant professor with the Department of Electrical Engineering at the University of Engineering and Technology, Jalozai Campus, Peshawar, Pakistan. He was a postdoctoral fellow within the Department of Electrical Engineering, Universidad de Chile, Santiago, between 2013 and 2015. His research interests include signal processing, pattern recognition, machine learning, and artificial intelligence.

Millaray Curilem (millaray.curilem@ufrontera.cl) received her B.E. degree in electrical engineering from the Instituto Superior Politécnico, José Antonio Echeverría, Cuba, in 1991 and her Ph.D. degree in electrical engineering from the Universidade Federal de Santa Catarina, Brazil, in 2002. She is a full professor with the Department of Electrical Engineering at the Universidad de La Frontera, Temuco, Chile. She was a postdoctoral fellow at the Universidad de Santiago de Chile in 2007 and at the Federal University of Bahia, Brazil, in 2017. At the Universidad de La Frontera, she is affiliated with the Signal Processing and Pattern Recognition Laboratory, where she researches machine-learning techniques applied to volcanology and biomedical problems. She is a member of the IEEE Signal Processing Society and the IEEE Computational Intelligence Society.

Fernando Huenupán (fernando.huenupan@ufrontera.cl) received his B. Sc. degree in electronic engineering from Universidad de La Frontera, Temuco, Chile, in 2004 and his Ph.D. degree in electrical engineering from the Univer-

sidad de Chile, Santiago, Chile, in 2010. He is an assistant professor with the Department of Electrical Engineering at the Universidad de La Frontera, Temuco, Chile. His research interests include signal processing and pattern analysis.

Muhammad Farhan Khan (farhan.khan@the-mtc.org) received his B.Sc., M.Sc., and Ph.D. degrees in mechanical engineering from the University of Engineering and Technology, Peshawar, Pakistan, in 2008, Loughborough University, United Kingdom, in 2010, and the University of Nottingham, United Kingdom, in 2018, respectively. He is currently affiliated with the Manufacturing Technology Centre, Coventry, United Kingdom. He worked as a research fellow in product and process development with the Institute of Innovation in Sustainable Engineering at the University of Derby, United Kingdom. Before joining the University of Derby, he worked on structural design optimization at the University of Nottingham, where he was a member of the additive manufacturing research group.

Néstor Becerra Yoma (nbecerra@ing.uchile.cl) received his B.Sc. and M.Sc. degrees from Campinas State University, São Paulo, Brazil, and his Ph.D. degree from the University of Edinburgh, United Kingdom, all in electrical engineering, in 1986, 1993, and 1998, respectively. From August 2016 to June 2017, he was a visiting professor within the Electric and Computer Engineering Department at Carnegie Mellon University, Pittsburgh, Pennsylvania. Since 2000, he has been a professor with the Department of Electrical Engineering, Universidad de Chile, Santiago, where he is currently lecturing on signal and speech processing, machine learning, and telecommunications. He served as an associate editor of *IEEE Transactions on Speech and Audio Processing* from 2009 to 2012. He is a Senior Member of the IEEE and a member of the International Speech Communication Association.

References

- [1] C. San-Martin, C. Melgarejo, C. Gallegos, G. Soto, M. Curilem, and G. Fuentealba, "Feature extraction using circular statistics applied to volcano monitoring," in *Proc. Iberoamerican Congr. Pattern Recognition*, 2010, pp. 458–466.

- [2] D. Gomez, M. Hellweg, B. Buttkus, F. Böker, M. L. Calvache, G. Cortés, E. Faber, F. Gil Cruz et al., "A volcano reawakens: Multiparameter observations of activity transition at Galeras Volcano (Colombia)," *Harvard*, 2004. [Online]. Available: <http://adsabs.harvard.edu/abs/2004AGUFM.V11B1423G>
- [3] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, Eds., *Feature Extraction: Foundations and Applications*. Berlin: Springer-Verlag, 2006.
- [4] M. Bicego, J. M. Londoño-Bonilla, and M. Orozco-Alzate, "Volcano-seismic events classification using document classification strategies," in *Proc. Int. Conf. Image Analysis and Processing*, 2015, pp. 119–129.
- [5] A. Esposito, F. Giudicepietro, L. D'Auria, S. Scarpetta, M. G. Martini, M. Coltelli, and M. Marinaro, "Unsupervised neural analysis of very-long-period events at Stromboli volcano using the self-organizing maps," *Bulletin Seismological Soc. America*, vol. 98, no. 5, pp. 2449–2459, 2008. doi: 10.1785/0120070110.
- [6] G. Curilem, J. Vergara, G. Fuentealba, G. Acuña, and M. Chacón, "Classification of seismic signals at Villarica volcano (Chile) using neural networks and genetic algorithms," *J. Volcanology Geothermal Res.*, vol. 180, no. 1, pp. 1–8, 2009. doi: 10.1016/j.jvolgeores.2008.12.002.
- [7] I. Alvarez, L. García, G. Cortes, C. Benítez, and Á. De la Torre, "Discriminative feature selection for automatic classification of volcano-seismic signals," *IEEE Geosci. Remote Sens. Lett.*, vol. 9, no. 2, pp. 151–155, 2012. doi: 10.1109/LGRS.2011.2162815.
- [8] G. Cortés, L. García, I. Álvarez, C. Benítez, Á. de la Torre, and J. Ibáñez, "Parallel system architecture (PSA): An efficient approach for automatic recognition of volcano-seismic events," *J. Volcanology Geothermal Res.*, vol. 271, pp. 1–10, Feb. 2014. doi: 10.1016/j.jvolgeores.2013.07.004.
- [9] M. Curilem, J. Vergara, C. San Martín, G. Fuentealba, C. Cardona, F. Huenupan, M. Chacón, M. S. Khan et al., "Pattern recognition applied to seismic signals of the Llaima volcano (Chile): An analysis of the events' features," *J. Volcanology Geothermal Res.*, vol. 282, pp. 134–147, Aug. 2014. doi: 10.1016/j.jvolgeores.2014.06.004.
- [10] K. Unglert, V. Radić, and A. M. Jellinek, "Principal component analysis vs. self-organizing maps combined with hierarchical clustering for pattern recognition in volcano seismic spectra," *J. Volcanology Geothermal Res.*, vol. 320, pp. 58–74, June 2016. doi: 10.1016/j.jvolgeores.2016.04.014.
- [11] G. Cortés, M. C. Benítez, L. García, I. Álvarez, and J. M. Ibanez, "A comparative study of dimensionality reduction algorithms applied to volcano-seismic signals," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 1, pp. 253–263, 2016. doi: 10.1109/JSTARS.2015.2479300.
- [12] R. Carniel, "Characterization of volcanic regimes and identification of significant transitions using geophysical data: A review," *Bulletin Volcanology*, vol. 76, pp. 848, Aug. 2014.
- [13] J. M. Ibáñez, C. Benítez, L. A. Gutiérrez, G. Cortés, A. García-Yeguas, and G. Alguacil, "The classification of seismo-volcanic signals using hidden Markov models as applied to the Stromboli and Etna volcanoes," *J. Volcanology Geothermal Res.*, vol. 187, no. 3–4, pp. 218–226, 2009. doi: 10.1016/j.jvolgeores.2009.09.002.
- [14] V. Jovivek, N. Chandrasekar, and Y. Srinivas, "Improving seismic monitoring system for small to intermediate earthquake detection," *Int. J. Comput. Sci. Security*, vol. 4, no. 3, pp. 308–315, 2010.
- [15] S. Scarpetta, F. Giudicepietro, E. C. Ezin, S. Petrosino, E. Del Pezzo, M. Martini, and M. Marinaro, "Automatic classification of seismic signals at Mt. Vesuvius volcano, Italy, using neural networks," *Bulletin Seismological Soc. America*, vol. 95, no. 1, pp. 185–196, 2005. doi: 10.1785/0120030075.
- [16] H. Langer, S. Falsaperla, T. Powell, and G. Thompson, "Automatic classification and a-posteriori analysis of seismic event identification at Soufrière Hills volcano, Montserrat," *J. Volcanology Geothermal Res.*, vol. 153, no. 1–2, pp. 1–10, 2006. doi: 10.1016/j.jvolgeores.2005.08.012.
- [17] A. Messina and H. Langer, "Pattern recognition of volcanic tremor data on Mt. Etna (Italy) with KAnalysis—A software program for unsupervised classification," *Comput. Geosci.*, vol. 37, no. 7, pp. 953–961, 2011. doi: 10.1016/j.cageo.2011.03.015.
- [18] F. U. Dowla, "Neural networks in seismic discrimination," in *Monitoring a Comprehensive Test Ban Treaty*, E. S. Husebye and A. M. Dainty, Eds. Dordrecht, The Netherlands: Springer, 1996, pp. 777–789. doi: 10.1007/978-94-011-0419-7_41.
- [19] P. Gendron and B. Nandram, "An empirical Bayes estimator of seismic events using wavelet packet bases," *J. Agricultural, Biological, Environmental Statist.*, vol. 6, pp. 379–402, Sept. 2001. doi: 10.1198/108571101317096587.
- [20] P. Lesage, F. Glangeaud, and J. Mars, "Applications of autoregressive models and time-frequency analysis to the study of volcanic tremor and long-period events," *J. Volcanology Geothermal Res.*, vol. 114, no. 3–4, pp. 391–417, 2002. doi: 10.1016/S0377-0273(01)00298-0.
- [21] G. Erlebacher and D. A. Yuen, "A wavelet toolkit for visualization and analysis of large data sets in earthquake research," *Pure Appl. Geophys.*, vol. 161, no. 11–12, pp. 2215–2229, 2004. doi: 10.1007/s00024-004-2559-5.
- [22] M. Beyreuther, C. Hammer, J. Wassermann, M. Ohrnberger, and T. Megies, "Constructing a hidden Markov model based earthquake detector: Application to induced seismicity," *Geophysical J. Int.*, vol. 189, no. 1, pp. 602–610, 2012. doi: 10.1111/j.1365-246X.2012.05361.x.
- [23] A. M. Esposito, F. Giudicepietro, S. Scarpetta, and S. Khilnani, "A neural approach for hybrid events discrimination at Stromboli volcano," in *Multidisciplinary Approaches to Neural Computing*, A. Esposito, M. Faudez-Zanuy, F. C. Morabito, and E. Pasero, Eds. Cham, Switzerland: Springer, 2017, pp. 11–21. doi: 10.1007/978-3-319-56904-8_2.
- [24] R. A. Altes, "Detection, estimation, and classification with spectrograms," *J. Acoust. Soc. Amer.*, vol. 67, no. 4, pp. 1232–1246, 1980. doi: 10.1121/1.384165.
- [25] M. Ibs-von Seht, "Detection and identification of seismic signals recorded at Krakatau volcano (Indonesia) using artificial neural networks," *J. Volcanology Geothermal Res.*, vol. 176, no. 4, pp. 448–456, 2008. doi: 10.1016/j.jvolgeores.2008.04.015.
- [26] B. Sick, M. Guggenmos, and M. Joswig, "Chances and limits of single-station seismic event clustering by unsupervised pattern recognition," *Geophysical J. Int.*, vol. 201, no. 3, pp. 1801–1813, 2015. doi: 10.1093/gji/ggv126.
- [27] C. Hibert, A. Mangeney, G. Grandjean, A. Peltier, A. DiMuro, N. M. Shapiro, V. Ferrazzini, P. Boissier et al., "Spatio-temporal evolution of rockfall activity from 2007 to 2011 at the Piton de la Fournaise volcano inferred from seismic data," *J. Volcanology Geothermal Res.*, vol. 333–334, pp. 36–52, Mar. 2017. doi: 10.1016/j.jvolgeores.2017.01.007.
- [28] P. Jousset, J. Neuberg, and S. Sturton, "Modeling the time-dependent frequency content of low-frequency volcanic earthquakes," *J. Volcanology Geothermal Res.*, vol. 128, no. 1–3, pp. 201–223, 2003.
- [29] R. Rekapalli, R. Tiwari, and M. Sen, "Fault identification by diffraction separation from seismic reflection data using time slice SSA-based algorithm," in *Proc. SEG Technical Program Expanded Abstracts 2016*, pp. 3920–3924.
- [30] M. Masotti, S. Falsaperla, H. Langer, S. Spampinato, and R. Campanini, "Application of support vector machine to the classification of volcanic tremor at Etna, Italy," *Geophys. Res. Lett.*, vol. 33, no. 20, pp. 1–5, 2006. doi: 10.1029/2006GL027441.
- [31] A. Köhler, M. Ohrnberger, and F. Scherbaum, "Unsupervised pattern recognition in continuous seismic wavefield records using self-organizing maps," *Geophysical J. Int.*, vol. 182, no. 3, pp. 1619–1630, 2010. doi: 10.1111/j.1365-246X.2010.04709.x.
- [32] R. Carniel, L. Barbui, and A. D. Jolly, "Detecting dynamical regimes by self-organizing map (SOM) analysis: An example from the March 2006 phreatic eruption at Raoul Island," *Bollettino Geofisica Teorica Applicata*, vol. 54, no. 1, pp. 39–52, 2013.
- [33] C. Cassisi, M. Prestifilippo, A. Cannata, P. Montalto, D. Patané, and E. Privitera, "Probabilistic reasoning over seismic time series: Volcano monitoring by hidden Markov models at Mt. Etna," *Pure Appl. Geophys.*, vol. 173, no. 7, pp. 2365–2386, 2016. doi: 10.1007/s00024-016-1284-1.
- [34] P. B. Quang, P. Gaillard, Y. Cano, and M. Ulzibat, "Detection and classification of seismic events with progressive multi-channel correlation and hidden Markov models," *Comput. Geosci.*, vol. 83, pp. 110–119, Oct. 2015. doi: 10.1016/j.cageo.2015.07.002.
- [35] A. M. Esposito, L. D'Auria, F. Giudicepietro, T. Caputo, and M. Martini, "Neural analysis of seismic data: Applications to the monitoring of Mt. Vesuvius," *Ann. Geophysics*, vol. 56, no. 4, 2013, Art. no. S0446.
- [36] M. Bicego, J. M. Londoño-Bonilla, and M. Orozco-Alzate, "Volcano-seismic events classification using document classification strategies," in *Proc. Int. Conf. Image Analysis and Processing*, 2015, pp. 119–129.
- [37] M. Titos, A. Bueno, L. García, and C. Benítez, "A deep neural networks approach to automatic recognition systems for volcano-seismic events," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 5, pp. 1533–1544, 2018. doi: 10.1109/JSTARS.2018.2803198.
- [38] M. Malfante, M. Dalla Mura, J.-P. Metaxian, J. I. Mars, O. Macedo, and A. Inza, "Machine learning for volcano-seismic signals: Challenges and perspectives," *IEEE Signal Process. Mag.*, vol. 35, no. 2, pp. 20–30, 2018. doi: 10.1109/MSP.2017.2779166.
- [39] M. Curilem, J. Vergara, C. San Martín, G. Fuentealba, C. Cardona, F. Huenupan, M. Chacón, M. S. Khan et al., "Pattern recognition applied to seismic signals of the Llaima Volcano (Chile): An analysis of the events' features," *J. Volcanology Geothermal Res.*, vol. 282, pp. 134–147, Aug. 2014. doi: 10.1016/j.jvolgeores.2014.06.004.
- [40] M. Curilem, F. Huenupan, C. San Martín, G. Fuentealba, C. Cardona, L. Franco, G. Acuña, and M. Chacón, "Feature analysis for the classification of volcanic seismic events using support vector machines," in *Proc. Mexican Int. Conf. Artificial Intelligence*, 2014, pp. 160–171.
- [41] M. Curilem, C. Soto, F. Huenupan, C. San Martín, C. Cardona, L. Franco, G. Acuña, M. Chacón et al., "Feature selection for discrimination between volcanic and tectonic events of the Llaima volcano (Chile)," in *Proc. Int. Conf. Pattern Recognition Systems*, 2016, pp. 1–5. doi: 10.1049/cic.2016.0034.
- [42] M. Curilem, C. Soto, F. Huenupan, C. San Martín, G. Fuentealba, C. Cardona, and L. Franco, "Improving the classification of volcanic seismic events extracting new seismic and speech features," in *Proc. Iberoamerican Congr. Pattern Recognition*, 2018, pp. 177–185. doi: 10.1007/978-3-319-75193-1_22.
- [43] M. Curilem, A. Cuevas, R. Soto, F. Huenupan, C. San Martín, M. S. Khan, F. Gil, C. Cardona et al., "Classification of volcanic seismic events: An expert knowledge analysis for feature selection," in *Proc. 8th Int. Conf. Pattern Recognition Systems*, 2017, pp. 2–6. doi: 10.1049/cp.2017.0131.
- [44] M. Curilem, J. P. Canario, L. Franco, and R. A. Rios, "Using CNN to classify spectrograms of

(continued on page 163)

Understanding the Basis of Graph Signal Processing via an Intuitive Example-Driven Approach

Graphs are irregular structures that naturally represent the multifaceted data attributes; however, traditional approaches have been established outside signal processing and largely focus on analyzing the underlying graphs rather than signals on graphs. Given the rapidly increasing availability of multisensor and multinode measurements, likely recorded on irregular or ad hoc grids, it would be extremely advantageous to analyze such structured data as “signals on graphs” and thus benefit from the ability of graphs to incorporate spatial sensing awareness, physical intuition, and sensor importance, together with the inherent “local versus global” sensor association. The aim of this lecture note is, therefore, to establish a common language between graph signals that are observed in irregular signal domains and some of the most fundamental paradigms in digital signal processing (DSP), such as spectral analysis, system transfer function, digital filter design, parameter estimation, and optimal denoising.

Scope

The move to gather more data from our environment, for our applications, health, and general well-being, is an established fact. The increase in the modalities of data acquisition and signal sensors, together with the corresponding increase in their structure and the complexity of information, has highlighted the need

for a shift in thinking and new analytical frameworks. This need becomes even clearer when we take into consideration the intermodality and interlocality attributes and their interactions, which effectively call for radically new data analytics approaches. Such a paradigm shift is provided by graph signal theory, a framework that goes beyond the standard “vertices, links, and their structural properties” components of a graph. It is the aim of this lecture note to introduce a unifying perspective based on a real-world multisensor problem.

This is achieved through a physically meaningful and intuitive real-world example of geographically distributed estimation of multisensor temperature measurements. A similar spatial multisensor arrangement has already been widely used in signal processing curricula to introduce minimum variance estimators and Kalman filters, and by adopting this framework, we facilitate a seamless integration of graph theory into the curriculum of existing DSP courses. By bridging the gap between standard approaches and graph signal processing, we also show that standard methods can be thought of as special cases of their graph counterparts, evaluated on path graphs. This article was, therefore, primarily written in response to the need to bring graph signal processing into the curricula of existing signal processing and machine learning courses, and this material corresponds to a 2-h lecture that could come at the very end of the standard lecture course syllabi. We also hope that our approach

will not only help to demystify graph-theoretic approaches for education purposes but also empower practitioners and researchers to explore a whole host of otherwise prohibitive modern applications. The supporting material, lecture slides, data, and MATLAB code can be found at <http://www.tfsa.ac.me/ln-gsp> and http://www.commsp.ee.ic.ac.uk/~mandic/DSP_ML_Education.htm.

Relevance

In classical signal processing, the signal domain is determined by equidistant time instants or by a set of spatial sensing points on a uniform grid. Increasingly, however, the actual data sensing domain may not even be related to the physical dimensions of time and/or space, and it typically exhibits various forms of irregularity, as, for example, in social or web-related networks, where the sensing points and their connectivity pertain to specific objects or nodes and the ad hoc topology of their links. It should be noted that even for the data acquired in well-defined time and space domains, the introduction of new relations between the signal samples, through graphs, may yield new insights into the analysis and provide enhanced data processing (for example, based on local similarity, through neighborhoods). We, therefore, set out to show that the advantage of graphs over classical data domains is that graphs account naturally and comprehensively for irregular data relations in the problem definition, together with the corresponding data connectivity in the analysis.

Through a real-world temperature estimation example, we show that graph signal and information processing is particularly well suited to making sense from data acquired over irregular data domains, which can be achieved, for example, by leveraging intuitions developed on Euclidean domains, by employing analogies with other irregular domains such as polygon meshes and manifolds, or by learning the mutual connectivity patterns from the available sets of data. In many emerging applications, for example, big data, this also introduces a number of new challenges:

- Basic concepts must be revisited to accommodate structured but often incomplete information.
- New physically meaningful frameworks, specifically tailored for heterogeneous data sources, are required.
- Tradeoffs between performance and numerical requirements are a prerequisite when operating in real time, especially given often combinatorial ways to analyze graphs.

The common language and enhanced intuition between the standard approaches and their graph counterparts, illuminated in this article through the relationships between the vertex and time domains, and between the corresponding eigenspectrum and frequency domains, may be naturally generalized to address the aforementioned challenges and spur further developments in the curricula on statistical signal processing, graph signal processing, and big data.

Prerequisites

This lecture note assumes a basic knowledge of linear algebra and DSP.

Problem statement and solutions

History of graph-theoretic applications

Graph theory, as a branch of mathematics, has existed for almost three centuries. The beginning of graph theory applications in electrical engineering dates back to the mid-19th century and the definition of Kirchhoff's laws. Owing to their inherent "spatial awareness," graph models have since become a de facto standard for data analytics across the science and

engineering areas, including chemistry, operational research, social networks, and computer sciences.

A systematic account of graph theory as an optimization tool can be attributed to the seminal book by Nicos Christofides of Imperial College London, published in 1975 [1]. Soon after graph theory gained prominence in general optimization, it was very natural to explore its application in signal processing and related data analytics areas [2]. Indeed, perhaps the first lecture course to teach graph theory to then emerging communication networks and channel coding student cohort was introduced by the author Anthony Constantinides in the 1970s. This helped to establish and formalize the connections between general optimization and the topology of a communication network and has spurred further applications in image processing [3].

After a relative lull in the field over more than two decades, current developments in graph theory owe their prominence to the emergence of modern data sources, such as large-scale sensor and social networks, which inherently provide rich underlying physical, social, and geographic structures. These require new ways to establish statistical inference, such as through data analytics on graphs and within a new, fast-maturing field of graph signal processing [4]–[10].

An illustrative example

Graph signal processing represents quite a general mathematical formalism that, while different from classic concepts, does admit the development of graph-domain counterparts of well-established DSP paradigms. It would, therefore, be quite valuable to introduce such a general concept in an inductive and intuitive way, through a simple yet general enough and well-understood example of a commonly considered topic in classical DSP.

To this end, we consider a multisensor setup, shown in Figure 1, for measuring temperature field in a known geographical region; such a setup is typically used in the context of minimum-variance estimation and Kalman filters. The temperature sensing locations are

chosen according to the significance of a particular geographic area to local users, with $N = 64$ sensing points in total, as shown in Figure 1(a). The temperature field is denoted by $\{x(n)\}$, and a snapshot of its values is given in Figure 1(b). Each such measured temperature signal can then be mathematically expressed as

$$x(n) = s(n) + \varepsilon(n), \quad (1)$$

where $s(n)$ is the true temperature that would have been obtained in ideal measuring conditions, while the term $\varepsilon(n)$ comprises the adverse effects of the local environment on sensor readings or faulty sensor activity and is referred to as *noise* in the sequel. For illustrative purposes, the noise $\varepsilon(n)$ was modeled in our study as a realization of white, zero-mean, Gaussian process, with standard deviation $\sigma_\varepsilon = 4$, that is $\varepsilon \sim \mathcal{N}(0, 16)$, which was added to the signal, $s(n)$, to yield the signal-to-noise ratio in $\{x(n)\}$ of $\text{SNR}_0 = 14.2$ dB.

Remark 1

Classical signal processing requires an arrangement of the quintessentially spatial temperature samples in Figure 1(a) into a line structure shown in Figure 1(c). Obviously, such "lexicographic" ordering is not amenable to exploiting the spatial information related to the actual sensor arrangement, dictated by the terrain. For example, this renders classical analyses of this temperature field inadequate (or at best suboptimal), as in this case the performance critically depends on the chosen sensor ordering scheme. This exemplifies that even a most routine temperature measurement setup requires a more complex estimation structure than the simple linear one that corresponds to the classical signal processing framework, as in Figure 1(c).

To introduce a "situation-aware" noise reduction scheme for the temperature field in Figure 1, we proceed to explore a graph-theoretic framework for this class of problems, starting from a local signal average operator. In classical signal processing, this can be achieved through a moving average operator, for example, through averaging across the neighboring

data samples, or equivalently neighboring nodes, as in the path graph in Figure 1(c), and for each sensing point. Physically, such a local neighborhood should indeed include geographically close neighboring sensing points, but these should also exhibit similar meteorological properties defined by the distance, altitude difference, and other terrain properties. In other words, since the sensor network in Figure 1 measures a set of related temperatures from irregularly spaced sensors, an effective estimation strategy should include domain knowledge, which is not possible to achieve with standard DSP (path graph).

Problem setup

Consider the local neighborhoods for the sensing points $n = 20, 29, 37$, and 41 , shown in Figure 2(a). The cumulative temperature for each sensing point is then given by

$$y(n) = \sum_{m \text{ at and around } n} x(m),$$

so that the local average temperature for a sensing point n may be easily obtained by dividing the cumulative temperature, $y(n)$, by the number of sensing points involved. For example, for the sensing points $n = 20$ and $n = 37$, presented in Figure 2(a), the “domain knowledge aware” local estimation takes the form

$$y(20) = x(20) + x(19) + x(22) + x(23) \quad (2)$$

$$y(37) = x(37) + x(32) + x(31) + x(35) + x(61). \quad (3)$$

For convenience, the full set of relations among the sensing points can now be rearranged into a matrix form, to give

$$\mathbf{y} = \mathbf{x} + \mathbf{Ax}, \quad (4)$$

where the matrix \mathbf{A} indicates the connectivity structure of the neighboring sensing locations that should be involved in the calculation for each estimate, $y(n)$. The matrix \mathbf{A} is therefore referred to as the *connectivity* or *adjacency matrix* of a graph. Its elements are either 1 (if the corresponding vertices are related) or 0 (if

they are not related). Figure 2(b) shows the sensing locations with the corresponding connectivity patterns for the temperature estimation scenario in Figure 2(a). From (2) we can observe, for example, that the 20th row of the adjacency matrix \mathbf{A} will have all zero elements, except for $A_{20,19} = 1$, $A_{20,22} = 1$, and $A_{20,23} = 1$

(see the supplementary materials in IEEE Xplore for more information).

This simple real-world example can be interpreted within the graph signal processing framework as follows:

- The sensing points where the signal is measured are designated as the *graph vertices* (see Figure 1).

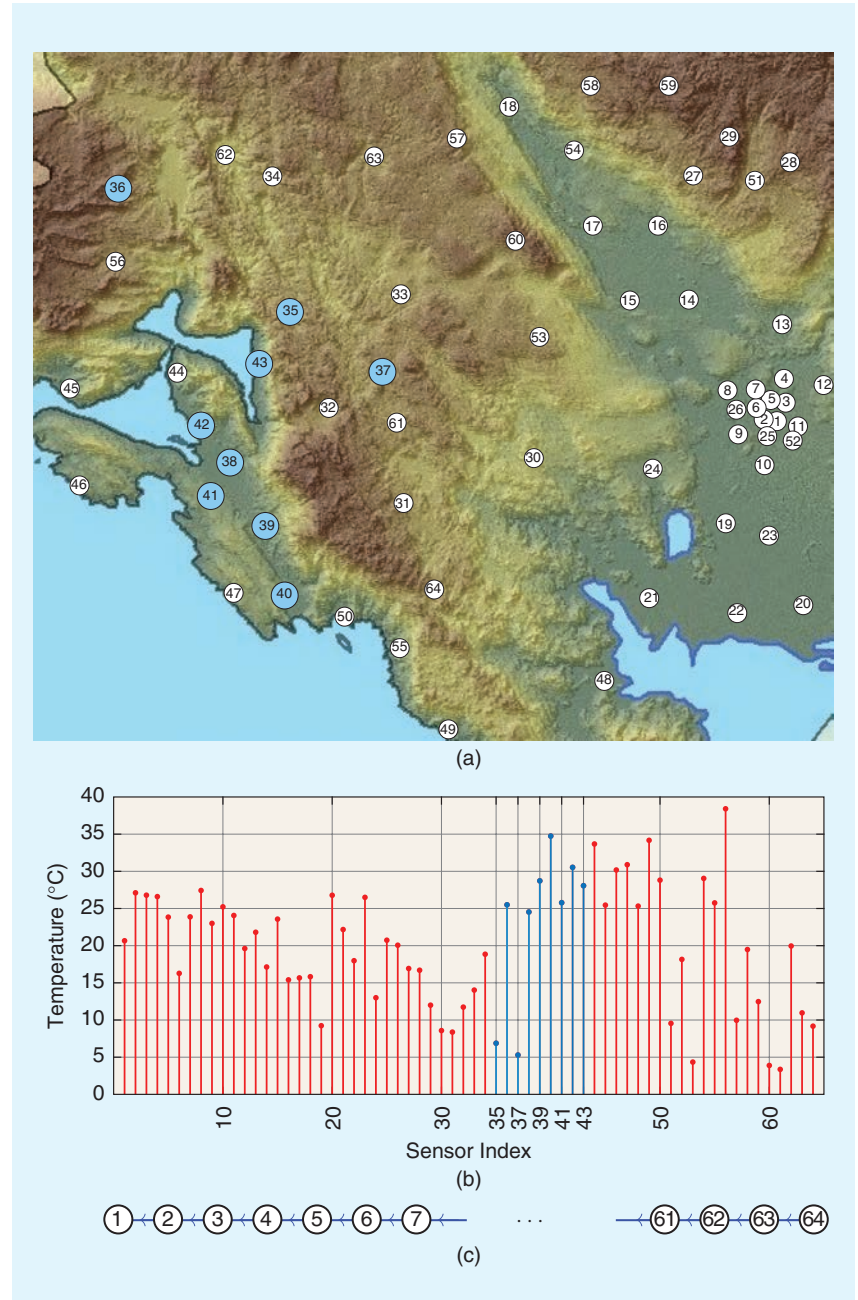


FIGURE 1. An illustration of temperature sensing as a classical signal processing problem. (a) A map of sensing locations in a geographic region along the Adriatic Sea (Montenegro). (b) Temperatures measured at $N = 64$ sensing locations. In standard signal processing, the spatial sensor index is used for the horizontal axis and serves as the signal domain. (c) This domain can also be interpreted as a directed path graph but lacks physical intuition as, for example, sensor 37 (mountains) is followed by sensor 38 (coast), with a drastic difference in temperature.

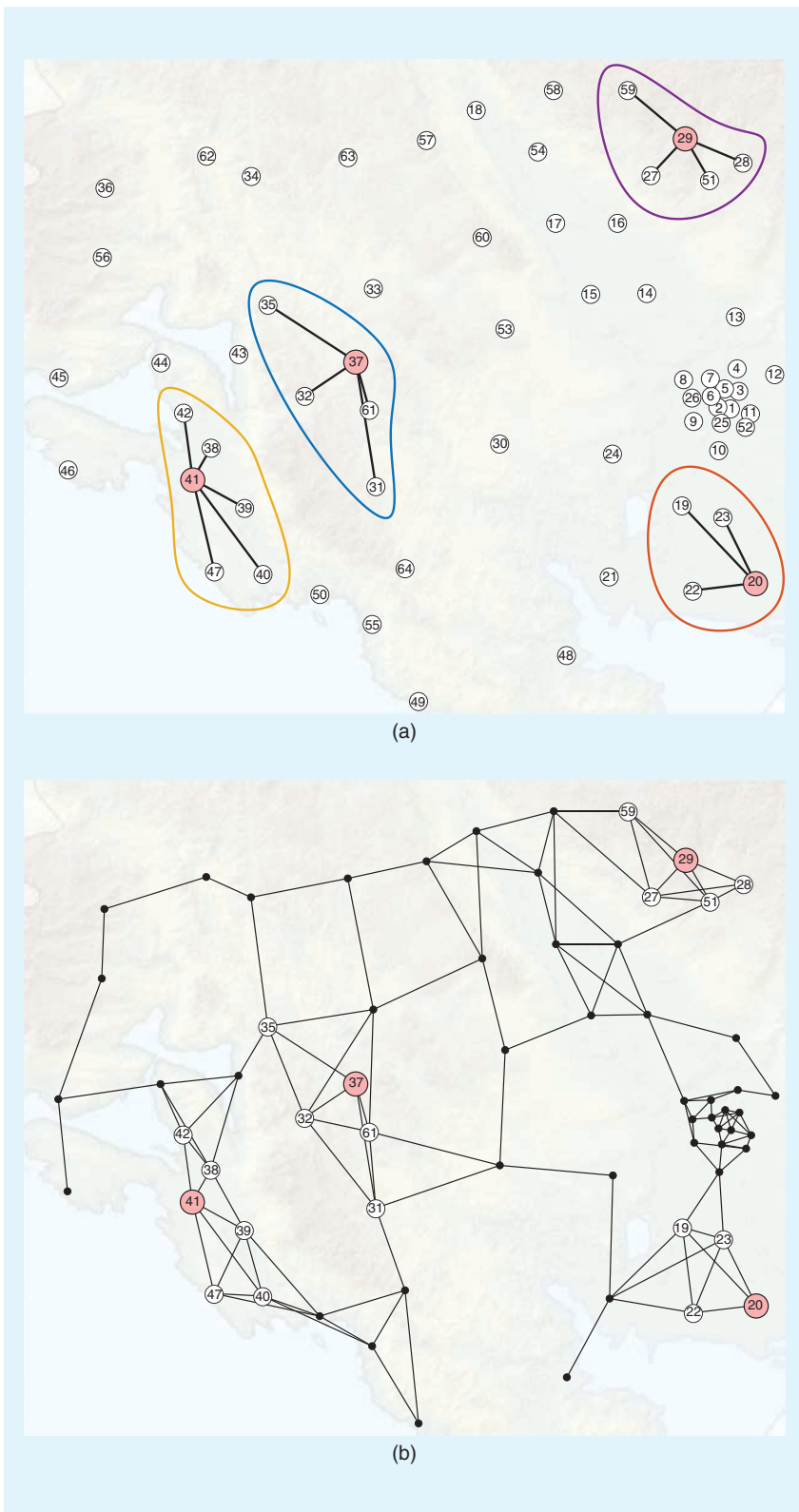


FIGURE 2. A temperature estimation setup as a domain-aware graph signal processing problem. (a) The local neighborhoods for the sensing points $n = 20, 29, 37$, and 41 . These neighborhoods are chosen using “domain knowledge,” dictated by local terrain and by taking into account the distance and altitude of sensors. This allows for the neighboring sensors for each of these sensing locations (vertices) to be chosen in a physically meaningful way, with their relation indicated by the connectivity lines, called *edges*. (b) The local neighborhoods for all sensing vertices from Figure 1, presented in a graph form.

- The vertex-to-vertex lines that indicate connectivity among the sensing points are called the *graph edges*.
- The vertices and edges form a *graph*, as in Figure 2(b), a new and very structurally rich signal domain.
- The graph, rather than a standard vector of sensing points, is then used for analyzing and processing data, as it is equipped with spatial and physical awareness.
- The measured temperatures are now interpreted as signal samples on a graph, as shown in Figure 3.
- Similar to traditional signal processing, this new graph signal may have many realizations on the same graph and may include noise, thus paving the way for statistical approaches.
- Through relation (4), we have, therefore, introduced a simple system for graph signals that performs physically and spatially aware signal averaging (a linear first-order system for a graph signal).

To emphasize our trust in a particular sensor and to model mutual sensor relevance, a weighting scheme may be imposed on the edges (connectivity) between the sensing points, in the form

$$y(n) = x(n) + \sum_{m \neq n} W_{nm} x(m). \quad (5)$$

The weight W_{nm} indicates the strength of the coupling between signal values at the sensing points n and m ; it has the value $W_{nm} = 0$ if the points n and m are not related or if $n = m$. We have now arrived at a weighted graph, whereby each edge has an associated weight, W_{nm} , which adds “mutual sensor relevance” information to the already established spatial awareness modeled by the edges. This equips graph signal models with additional flexibility. In our example, a matrix form of a weighted cumulative graph signal now becomes

$$\mathbf{y} = \mathbf{x} + \mathbf{W}\mathbf{x}. \quad (6)$$

To produce unbiased estimates, instead of the cumulative sums in (4) and (5), the weighting coefficients within the estimate for each $y(n)$ should sum up to

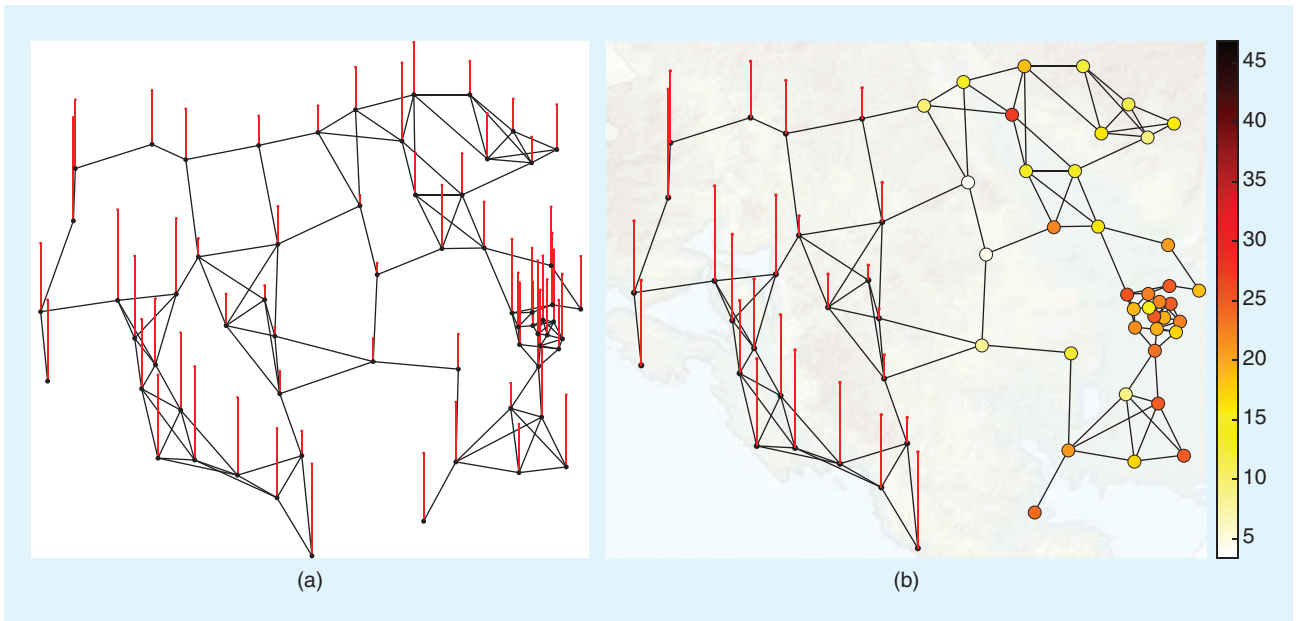


FIGURE 3. The move from a multisensor measurement to a graph signal. (a) The temperature field is represented on a graph that combines spatially unaware measurements from Figure 1(b) and the physically relevant graph topology from Figure 2(b). (b) The graph signal intensity may also be designated by the vertex color, as in the right half of the panel.

unity. This may be achieved through a normalized form of (6), given by

$$\mathbf{y} = \frac{1}{2}(\mathbf{x} + \mathbf{D}^{-1}\mathbf{W}\mathbf{x}), \quad (7)$$

where the elements of the diagonal normalization matrix, \mathbf{D} , called the *degree matrix*, are $D_{nn} = \sum_m W_{nm}$ while the term $\mathbf{D}^{-1}\mathbf{W}$ is referred to as a *random walk or diffusion weight matrix*. When this simple normalized first-order system is employed to filter the original noisy signal from Figure 3, the SNR = 20.2 dB was achieved—an improvement of 6 dB over the original signal-to-noise ratio, $\text{SNR}_0 = 14.2$ dB.

Another important operator for graph signal processing is the graph Laplacian, \mathbf{L} , which is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}.$$

Remark 2

A graph is fully specified by the set of its vertices and their connectivity scheme (designated by edges). The edges may be defined by the adjacency matrix, \mathbf{A} , with $A_{mn} \in \{0, 1\}$, for unweighted graphs or by the “connectivity strength” matrix, referred to as the *weight matrix*, \mathbf{W} , with $W_{mn} \in \mathbb{R}^+$, for weighted graphs.

The degree matrix, \mathbf{D} , and the Laplacian matrix, \mathbf{L} , with $L_{mn} \in \mathbb{R}$, are defined using the adjacency/weight matrix. When the relations between all pairs of vertices are mutually symmetric, then all the matrices involved are also symmetric, and such graphs are called *undirected*. If that is not the case, then the adjacency/weight matrix is not symmetric, and such graphs are called *directed graphs*.

The previously introduced graph framework is quite general and admits application to many different scenarios. For example, when performing an opinion poll within a social network, the members of that social network are treated as vertices (data acquisition points), while their friendship relations are represented by the edges that model graph connectivity, with the member attributes playing the role of graph signal values.

Remark 3

The definition of an appropriate graph structure is a prerequisite for physically meaningful and computationally efficient graph signal processing applications. Three important classes of problems, regarding the definition of the graph topology, are described in “Graph Topology (Edges and Weights).” In the following, we demonstrate how this

simple and intuitive concept provides a natural and straightforward platform to introduce the graph counterparts of several fundamental signal processing algorithms.

System for graph signals

In classical signal processing, a system is an operator that transforms an input signal into another (output) signal. The purpose of this section is to provide a definition of a *system that operates over graph signals*. Our focus will be on a subclass of systems called *graph shift invariant systems*, also referred to by some authors as simply *graph filters*. As illustrated in the following pages, this class of systems accounts for the topology of the graph where the signals reside, while maintaining analytical and computational tractability.

Signal shift on a graph

The signal shift operator (unit time delay) is the linchpin in discrete-time signal processing, but its definition on graphs is not so obvious due to the rich underlying connectivity structure. Topologically, the signal shift on a graph can be viewed as the movement of a signal sample from the considered vertex along all edges connected to this vertex. The (backward)

shift operator on a graph can then be compactly defined using the graph adjacency matrix as $\mathbf{x}_{\text{shifted}} = \mathbf{A}\mathbf{x}$.

To draw distinction between the standard shift and the graph shift operator, consider the path graph in Figure 1(c) and the spatially aware graph in Fig-

ure 2, and assume that the input signal is a pulse that occurs only at the sensor $n = 29$, that is, $x(n) = \delta(n - 29)$. The shifted signal in classical signal processing [path graph in Figure 1(c)] will be $x_{\text{shifted}}(n) = \delta(n - 28)$ and can be considered as a movement of the

delta pulse along the path graph from vertex n to vertex $(n - 1)$. The same principle can be applied to the graph domain in Figure 2(a), whereby the delta pulse from vertex $n = 29$ is moved to all its connected vertices, to obtain the shifted graph signal,

Graph Topology (Edges and Weights)

While in classical graph theory, the graphs are typically given (e.g., in various computer, social, road, transportation, and power networks), in graph signal processing oftentimes the first step is to employ background knowledge of signal-generating mechanisms to define the graph as a signal domain. This poses a number of challenges; e.g., while the data sensing points (graph vertices) are usually well defined in advance, their connectivity (graph edges) is often not available. In other words, the data domain definition within the graph signal paradigm represents a part of the problem itself and has to be determined based on the properties of the sensing positions or features of the acquired set of data. All in all, the definition of an appropriate graph structure is a prerequisite for physically meaningful and computationally efficient graph signal processing applications.

Three important classes of problems regarding the definition of graph edges are as follows:

- *Geometry of the vertex positions:* The distances between vertex positions play a crucial role in establishing relations among the sensed data. In many physical processes, both the existence of edges and their associated connecting weights are defined based on vertex distances. To this end, an exponential function of the

Euclidean distance between vertices, r_{mn} , may be used, where for a given distance threshold, τ ,

$$W_{mn} = e^{-r_{mn}^2/\alpha} \quad \text{or} \quad W_{mn} = e^{-r_{mn}/\alpha} \quad \text{if } r_{mn} < \tau,$$

and $W_{mn} = 0$ for $r_{mn} \geq \tau$. This form has been used in the graph in Figure 2, whereby the altitude difference, h_{mn} , was accounted for as $W_{mn} = e^{-r_{mn}/\alpha} e^{-h_{mn}/\beta}$.

- *Physically well-defined relations among the sensing positions:* Examples include electric circuits, linear heat transfer systems, spring-mass systems, and various forms of networks such as social, computer, or power networks. In these cases, the edge weights are given as a part of problem definition.
- *Data similarity dictates graph topology:* This scenario is the most common in image and biomedical signal processing (see "Graph Topology Based on Signal Similarity: An Image Processing Example"). Various approaches and metrics can be used to define data similarity, including the correlation matrix between the signals at various vertices or the corresponding inverse covariance (precision) matrix, combined with signal smoothness and edge sparsity conditions.

Learning a graph (its edges) based on the set of the available data is an interesting and currently extensively studied research area.

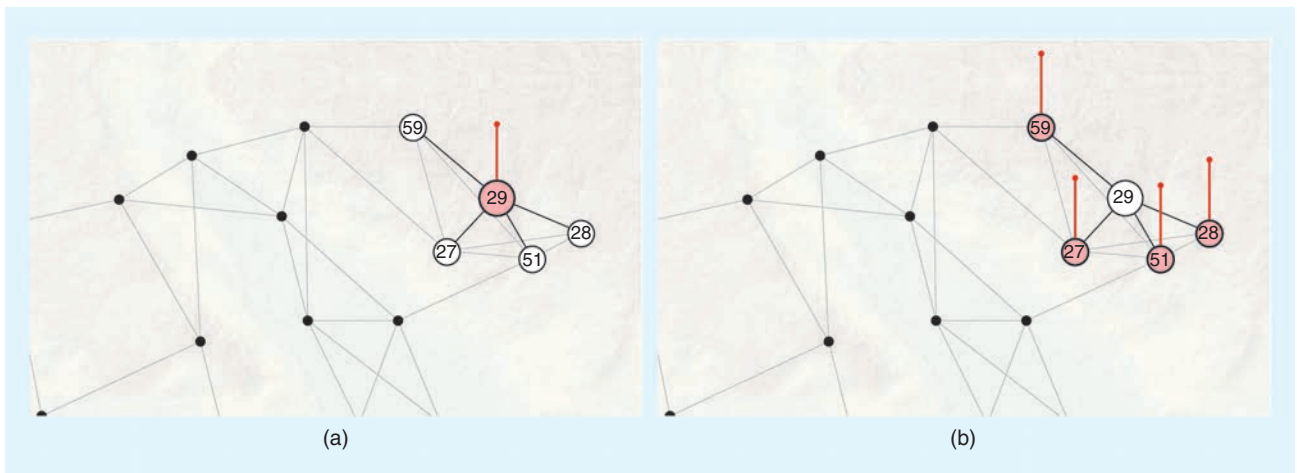


FIGURE 4. A shift operator on a graph. (a) A single-pulse graph signal, \mathbf{x} , located at the vertex $n = 29$, and given by $x(n) = \delta(n - 29)$. (b) The graph shifted version of \mathbf{x} , given by $\mathbf{x}_{\text{shifted}} = \mathbf{A}\mathbf{x}$. The graph shift operator is demonstrated on the northeast part of the graph from Figure 3, around the vertex $n = 29$.

$x_{\text{shifted}}(n) = \delta(n - 27) + \delta(n - 28) + \delta(n - 51) + \delta(n - 59)$, as shown in Figure 4.

If the values of the shifted signal are also scaled by the weighting coefficients of the corresponding edges, then the shifted signal is given by $\mathbf{x}_{\text{shifted}} = \mathbf{W}\mathbf{x}$. Observe that the Laplacian operator applied on a signal, $\mathbf{L}\mathbf{x}$, can also be considered as a graph shift operator, since it is a combination of the scaled original signal, $\mathbf{D}\mathbf{x}$, and its weighted shifted version, $\mathbf{W}\mathbf{x}$, that is, $\mathbf{L}\mathbf{x} = \mathbf{D}\mathbf{x} - \mathbf{W}\mathbf{x}$. In the sequel, we will adopt the symbol \mathbf{S} to denote a general shift operator on a graph, which yields a graph shifted signal, $\mathbf{x}_{\text{shifted}} = \mathbf{S}\mathbf{x}$.

Remark 4

The standard shift operator, $x(n) = x(n - 1)$, is a “one-to-one” mapping, while the graph shift operator, $\mathbf{x}_{\text{shifted}} = \mathbf{S}\mathbf{x}$, is a “one-to-many” mapping, which accounts for the underlying physics of the sensing process (as in our example), not possible to achieve with standard DSP. Moreover, by its very nature, the graph shift also allows us to incorporate a contextual relation between the vertices within an irregular grid through the weight matrix \mathbf{W} . Notice that the above graph shift operator does not satisfy the isometry property, since the energy of the shifted signal is not the same as the energy of the original signal.

System for graph signals

In analogy with the pivotal role of time shift in standard system theory, a system for graph signals can be implemented as a linear combination of a graph signal and its graph shifted versions. Here, the notion of a system is used in its classical sense, as a set of physical rules (an algorithm) that transforms an input graph signal into another (output) graph signal. The output graph signal can then be written as

$$\mathbf{y} = h_0 \mathbf{S}^0 \mathbf{x} + h_1 \mathbf{S}^1 \mathbf{x} + \dots + h_{M-1} \mathbf{S}^{M-1} \mathbf{x} = \sum_{m=0}^{M-1} h_m \mathbf{S}^m \mathbf{x}, \quad (8)$$

where, by definition, $\mathbf{S}^0 = \mathbf{I}$, while h_0, h_1, \dots, h_{M-1} are the system coefficients

(see the section “Spectral Domain Graph Filter Design”).

Remark 5

Common choices for the graph shift operator are 1) the adjacency matrix, $\mathbf{S} = \mathbf{A}$, and 2) graph Laplacian, $\mathbf{S} = \mathbf{L}$. Normalized versions of the adjacency matrix, graph Laplacian, $\mathbf{S} = \mathbf{D}^{-1/2} \mathbf{L} \mathbf{D}^{-1/2}$, and random walk (diffusion) matrix, $\mathbf{S} = \mathbf{D}^{-1} \mathbf{W}$, can also be used.

Notice that for the directed and unweighted path graph in Figure 1(c), the system for graph signals in (8) reduces to the well-known standard finite impulse response filter, given by

$$y(n) = h_0 x(n) + h_1 x(n - 1) + \dots + h_{M-1} x(n - M + 1). \quad (9)$$

Remark 6

The previously established link between the classical transfer function of a physical system and its graph-theoretic counterpart may serve to promote new algorithmic approaches that stem from signal processing into many application scenarios that may be related to graphs.

Remark 7

A system defined based on the graph Laplacian, \mathbf{L} , is obtained from (8) by replacing $\mathbf{S} = \mathbf{L}$ and allows us to produce an unbiased estimate of a constant, c , whereby if $\mathbf{x} = \mathbf{c}$ then $\mathbf{y} = \mathbf{c}$, since by the definition of the graph Laplacian, $\mathbf{L}\mathbf{c} = \mathbf{0}$. Hence, a simple first-order system based on the graph Laplacian can be written as

$$\mathbf{y} = \mathbf{x} + h_1 \mathbf{L}\mathbf{x} \quad (10)$$

and is amenable to being used for efficient low-pass graph filtering (see “Smoothness and Filtering on a Graph”).

Remark 8

A system for graph signals is conveniently defined by the graph transfer function, $H(\mathbf{S})$, as

$$\mathbf{y} = H(\mathbf{S})\mathbf{x}. \quad (11)$$

Properties of a system for graph signals

Following the aforementioned discussion, it is now possible to define the properties of systems for graph signals. From (8)–(11), the system for graph signals is said to be:

■ linear, if

$$H(\mathbf{S})(a_1 \mathbf{x}_1 + a_2 \mathbf{x}_2) = a_1 \mathbf{y}_1 + a_2 \mathbf{y}_2$$

■ shift invariant, if

$$H(\mathbf{S})(\mathbf{S}\mathbf{x}) = \mathbf{S}(H(\mathbf{S})\mathbf{x}).$$

Remark 9

A system for graph signals, defined as in (8), in the form

$$H(\mathbf{S}) = h_0 \mathbf{S}^0 + h_1 \mathbf{S}^1 + \dots + h_{M-1} \mathbf{S}^{M-1} \quad (12)$$

is linear and shift invariant, since the matrix multiplication of the square shift matrices is associative ($\mathbf{S}(\mathbf{S}\mathbf{S}) = (\mathbf{S}\mathbf{S})\mathbf{S}$), that is, $\mathbf{S}\mathbf{S}^m = \mathbf{S}^m \mathbf{S}$.

Graph Fourier transform

While classic spectral analysis is performed in the Fourier domain, spectral representations of graph signals employ the eigenspectrum (or simply “spectrum” hereafter) of the graph shift operator, \mathbf{S} , given by

$$\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1},$$

where \mathbf{U} is an orthonormal matrix of the eigenvectors, \mathbf{u}_k , in its columns, and $\mathbf{\Lambda}$ is a diagonal matrix of the corresponding eigenvalues, λ_k .

As in the majority of the literature dealing with the analysis of the frequency content of graph signals, here we will use $\mathbf{S} = \mathbf{L}$ in numerical examples, while $\mathbf{S} = \mathbf{A}$ is also used in illustrative examples, especially when relating graph signal processing to classical signal processing. The eigenvectors of graph Laplacian, \mathbf{L} , may then be used for the spectral-based clustering of graph vertices, as shown in “Vertex Clustering.”

The graph Fourier transform (GFT), \mathbf{X} , of a graph signal, \mathbf{x} , is then defined as

$$\mathbf{X} = \mathbf{U}^{-1} \mathbf{x}. \quad (13)$$

Smoothness and Filtering on a Graph

The quadratic form of a graph signal on an undirected graph is given by

$$E_x = \mathbf{x}^T \mathbf{L} \mathbf{x} = \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N W_{nm} (x(n) - x(m))^2$$

and can be used to define signal smoothness, since small values of the squared local deviation, $(x(n) - x(m))^2$, correspond to a smooth, slow-varying signal. For a constant signal, $\mathbf{x} = \mathbf{c}$, we, therefore, have $E_x = 0$.

Physically, the minimum of $\mathbf{x}^T \mathbf{L} \mathbf{x}$ implies the smoothest possible signal, and to arrive at this solution we may employ steepest descent. Then, the signal value at an iteration p is adjusted in the opposite direction of the gradient, toward the minimum of $\mathbf{x}^T \mathbf{L} \mathbf{x}$. The gradient of this quadratic form is $\partial E_x / \partial \mathbf{x}^T = 2\mathbf{L} \mathbf{x}$, which yields the iterative procedure

$$\mathbf{x}_{p+1} = \mathbf{x}_p - \alpha \mathbf{L} \mathbf{x}_p = (\mathbf{I} - \alpha \mathbf{L}) \mathbf{x}_p.$$

Notice that the signal \mathbf{x}_{p+1} can be considered as an output of the first-order system in (11), with $h_1 = -\alpha$, and this relation can be used for simple and efficient filtering of graph signals.

Since the minimum of the quadratic form $\mathbf{x}^T \mathbf{L} \mathbf{x}$ corresponds to a constant signal, to avoid obtaining only a constant steady state (that is, to also account for the slow-varying part of the graph signal), the aforementioned iteration process can be used in alternation with $\mathbf{x}_{p+2} = (\mathbf{I} + \beta \mathbf{L}) \mathbf{x}_{p+1}$. A compact form of these two iterative processes is known as Taubin's $\alpha - \beta$ algorithm and is given by

$$\mathbf{x}_{p+2} = (\mathbf{I} + \beta \mathbf{L})(\mathbf{I} - \alpha \mathbf{L}) \mathbf{x}_p. \quad (\text{S1})$$

For appropriate values of α and β , this system can give a good and very simple approximation of a low-pass graph filter with transfer function $H(\lambda_k) = (1 + (\beta - \alpha)\lambda_k - \alpha\beta\lambda_k^2)^P$, and in P iterations, where k denotes the spectral index (see the section "Spectral Domain Graph Filter Design").

In our experiment, the original noisy signal from Figure 3 was filtered using Taubin's algorithm, with $\alpha = 0.2$ and

$\beta = 0.1$. After 50 iterations, the signal-to-noise ratio improved from the original $\text{SNR}_0 = 14.2$ dB to 26.8 dB (see Figure S1). With these parameters, the transfer function, $H(\lambda_k)$, retained seven out of 64 spectral components in the signal (with an attenuation lower than 3 dB).

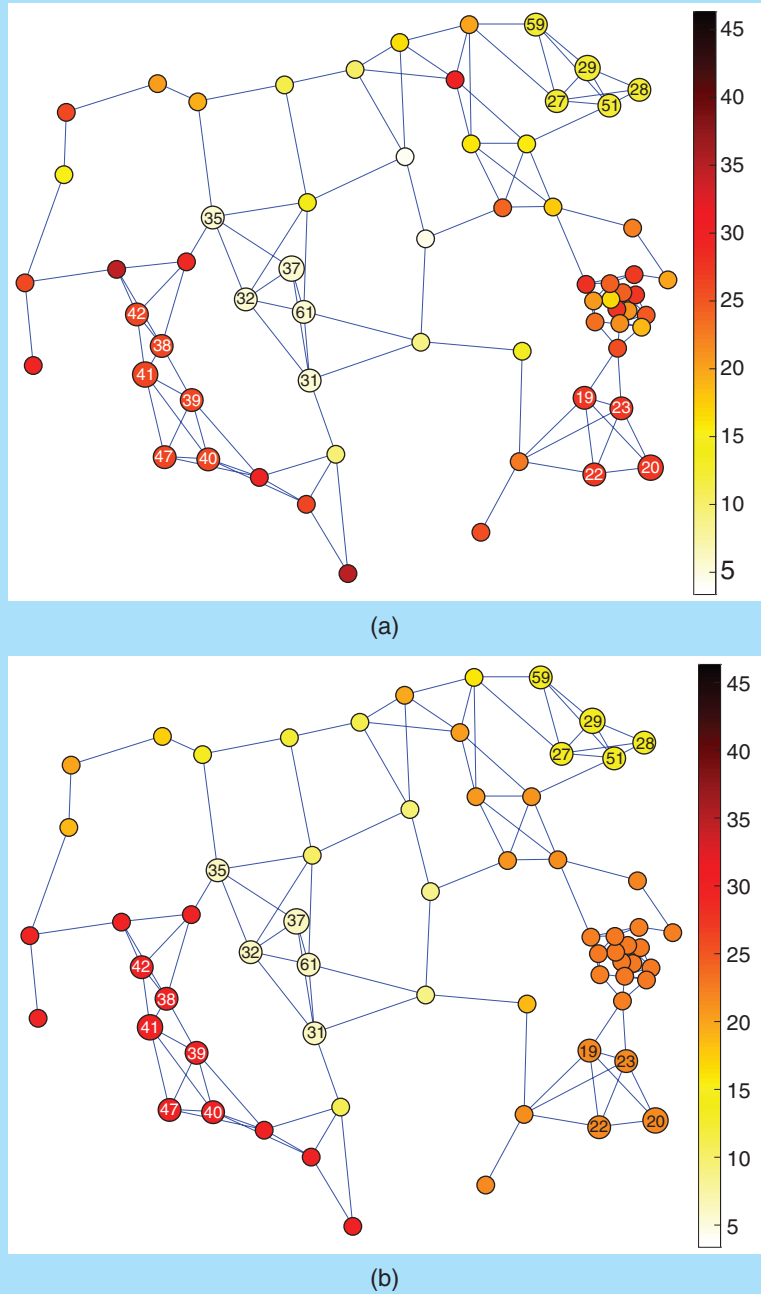


FIGURE S1. An illustration of low-pass filtering on a graph. (a) The original noisy signal. (b) The filtered signal. The graph signal intensity is designated by the vertex color.

Vertex Clustering

The term *vertex clustering* here refers to the task of identifying and arranging the vertices of a graph into nonoverlapping vertex subsets, with data in each subset expected to exhibit relative similarity in some sense. One efficient approach to vertex clustering is based on spectral graph analysis. For a graph with N vertices, the orthogonal eigenvectors of its Laplacian build an N -dimensional space, called the *spectral space*. The elements $u_k(n)$ of the eigenvector \mathbf{u}_k , $k=1, 2, \dots, N$, can then be assigned to vertices n , $n=1, 2, \dots, N$ to form an N -dimensional spectral vector $\mathbf{q}_n = [u_1(n), u_2(n), \dots, u_N(n)]$. The elements of the first eigenvector, \mathbf{u}_1 , of the graph Laplacian are constant and are omitted, since they do not convey any spectral difference to the graph vertices.

For the purpose of vertex clustering, the original N -dimensional spectral vector space may be reduced to a new L -dimensional spectral space ($L < N$), where the spectral vectors,

$$\mathbf{q}_n = [u_2(n), u_3(n), \dots, u_{L+1}(n)],$$

are used to define spectral similarity between any two vertices, n and m , as $\|\mathbf{q}_n - \mathbf{q}_m\|_2$. Vertex clustering is then performed by grouping spectrally similar vertices.

The simplest (and most widely used) case is when only one eigenvector, \mathbf{u}_2 , is used for spectral clustering, whereby the order of vertices in the sorted \mathbf{u}_2 corresponds to its smoothest representation. This procedure can be used for ordering the vertices in graphs, even if we desire to perform any form of classical presentation or processing with vertices on a path graph, as in Figure 1(c).

The spectral vector, \mathbf{q}_n , can be either used to designate a position of a vertex in a new low L -dimensional space, or it can be used for coloring of the vertices at their original positions. For the graph from Figure 2, such coloring was performed using the spectral vector elements $\mathbf{q}_n = [u_2(n), u_3(n), u_4(n)]$ as color coordinates for

the vertex n (see Figure S2). Similar colors indicate high spectral similarity.

Note that vertex clustering is a signal-independent operation. It just roughly indicates the expected relation between sensor data values on the considered graph and suggests that data processing operations (including processing of the signal from Figure 3) will be predominantly localized within these clusters.

Formally, the so-achieved reduction in spectral vertex dimensionality, from the original N eigenvectors to L eigenvectors with lowest variations (with the smallest smoothness index $\mathbf{u}_k^T \mathbf{L} \mathbf{u}_k = \lambda_k$), corresponds to low-pass filtering in graph signal processing, whereby a signal with N spectral components is projected onto a reduced spectral space with L slowest-varying spectral components, within a given set of basis functions (cf. truncated Fourier representation).

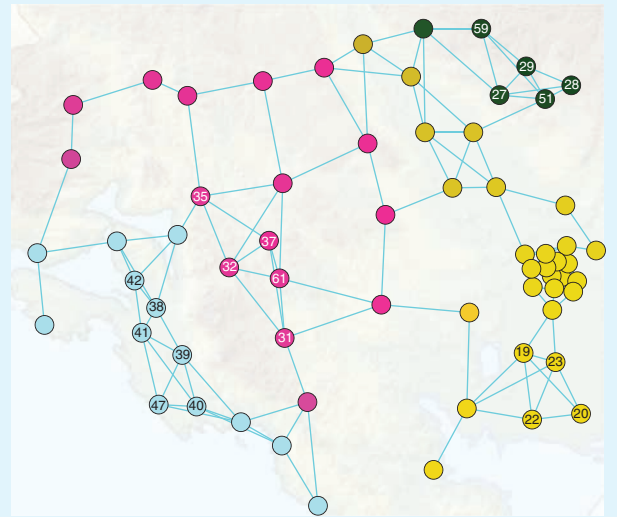


FIGURE S2. The vertices in the graph from Figure 2 have been colored using the spectral vectors $\mathbf{q}_n = [u_2(n), u_3(n), u_4(n)]$ as red, green, blue color coordinates.

The *GFT domain* is referred to as the *graph spectral domain*, since the domain for the GFT, \mathbf{X} (with elements commonly denoted by $X(k)$ or $X(\lambda_k)$), is the graph spectrum, λ_k , $k=1, 2, \dots, N$.

Physically, since $\mathbf{U}^{-1} = \mathbf{U}^T$, the element $X(k)$ of a GFT, \mathbf{X} , represents a projection of the graph signal, \mathbf{x} , onto the k th eigenvector, $\mathbf{u}_k \in \mathbf{U}$, that is

$$X(k) = \sum_{n=1}^N x(n) u_k(n). \quad (14)$$

The inverse GFT is then straightforwardly obtained as

$$\mathbf{x} = \mathbf{U} \mathbf{X} \quad (15)$$

or

$$x(n) = \sum_{k=1}^N X(k) u_k(n). \quad (16)$$

Remark 10

In analogy to the classical Fourier transform where the signal is projected onto a set of harmonic orthogonal bases, $\mathbf{X} = \mathbf{U}^{-1} \mathbf{x}$,

where \mathbf{U} is the matrix of harmonic bases, $\mathbf{u}_k = [1, e^{j2\pi(k-1)/N}, \dots, e^{j2\pi(N-1)(k-1)/N}]^T / \sqrt{N}$, the GFT can be understood as a signal decomposition onto the set of eigenvectors of the graph Laplacian (or the adjacency matrix) that serve as orthogonal basis functions. In the case of a directed circular graph, the GFT reduces to the standard discrete Fourier transform (DFT). For this reason, the transform in (14) is referred to as the GFT.

Classic spectral analysis can thus be considered as a special case of graph signal

spectral analysis, with the adjacency matrix defined on an unweighted circular directed graph (a path graph with the connected last and first vertex), where $\mathbf{u}_k = [1, e^{j2\pi(k-1)/N}, \dots, e^{j2\pi(N-1)(k-1)/N}]^T / \sqrt{N}$. This becomes obvious upon recognizing that the eigenvalues of a directed unweighted circular graph, $\lambda_k = e^{-j2\pi(k-1)/N}$, are easily obtained as a solution of the eigenvalue/eigenvector relation $\mathbf{A}\mathbf{u}_k = \lambda_k \mathbf{u}_k$. For a vertex n , this relation is of the form $u_k(n-1) = \lambda_k u_k(n)$. The solutions of this difference equation are the elements of the previously discussed eigenvector, $u_k(n) = e^{j2\pi(k-1)(n-1)/N} / \sqrt{N}$, and the corresponding eigenvalues, $\lambda_k = e^{-j2\pi(k-1)/N}$. It can be shown that the eigenvectors of the graph Laplacian of a circular graph are real-valued harmonic functions, whose combinations can produce the standard complex-valued DFT basis functions, albeit in an indirect way. The standard signal representation in Figure 1(b), therefore, corresponds to a signal whose domain is a path graph.

As is common in signal processing, for our example in Figure 1 and (1) the temperature values were generated through a linear combination of several graph Laplacian eigenvectors (serving as basis functions) in the form $\mathbf{x} = 160\mathbf{u}_1 + 16\mathbf{u}_2 - 8\mathbf{u}_3 - 40\mathbf{u}_4 + 16\mathbf{u}_5 - 24\mathbf{u}_6 + \varepsilon(n)$, where the random Gaussian noise, $\varepsilon(n)$, had standard deviation $\sigma_\varepsilon = 4$,

to yield the signal-to-noise ratio of $\text{SNR}_0 = 14.2$ dB.

Spectral domain of a system for graph signals

Consider a system for graph signals, as in (8), defined by a general shift operator on a graph \mathbf{S} , given by

$$\mathbf{y} = \sum_{m=0}^{M-1} h_m \mathbf{S}^m \mathbf{x}. \quad (17)$$

Upon employing the eigendecomposition of the shift operator, $\mathbf{S} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^{-1}$, we arrive at

$$\mathbf{y} = \sum_{m=0}^{M-1} h_m \mathbf{U} \mathbf{\Lambda}^m \mathbf{U}^{-1} \mathbf{x} = \mathbf{U} H(\mathbf{\Lambda}) \mathbf{U}^{-1} \mathbf{x}, \quad (18)$$

where

$$H(\mathbf{\Lambda}) = \sum_{m=0}^{M-1} h_m \mathbf{\Lambda}^m \quad (19)$$

is the transfer function of the system for graph signals.

From (18), $\mathbf{U}^{-1} \mathbf{y} = H(\mathbf{\Lambda}) \mathbf{U}^{-1} \mathbf{x}$, or in terms of the GFT of the input and output signal

$$\mathbf{Y} = H(\mathbf{\Lambda}) \mathbf{X}. \quad (20)$$

The classic spectral transfer function for (9) is then obtained by using the adjacency matrix of an unweighted directed circular graph, whose eigenvalues are $\lambda_k = e^{-j2\pi(k-1)/N}$.

Spectral domain graph filter design

A system for graph signals that is designed to modify spectral content of graph signals in a desired way shall be referred to as a *graph filter*. Consider a graph filter with a desired transfer function, $G(\mathbf{\Lambda})$. As in classical signal processing, a filter with this transfer function can be implemented either in the spectral domain or in the vertex domain.

The spectral domain implementation is straightforward and can be performed in the following three steps:

- Calculate the GFT of the input graph signal, \mathbf{x} , in the form $\mathbf{X} = \mathbf{U}^{-1} \mathbf{x}$.
- Multiply the GFT of \mathbf{x} with the transfer function, $G(\mathbf{\Lambda})$, to obtain $\mathbf{Y} = G(\mathbf{\Lambda}) \mathbf{X}$.
- Calculate the output graph signal as the inverse GFT of \mathbf{Y} , to yield $\mathbf{y} = \mathbf{U} \mathbf{Y}$.

Notice that this procedure may be computationally very demanding for large graphs, where it may be easier to implement the desired filter (or its close approximation) in the vertex domain, in analogy to the time domain in the classical approach. In other words, we have to find the coefficients, h_0, h_1, \dots, h_{M-1} in (8), such that their spectral representation, $H(\mathbf{\Lambda})$, is equal (or at least as close as possible) to the desired graph filter $G(\mathbf{\Lambda})$, and then to implement the system in the vertex domain using (17).

The previously mentioned condition that the transfer function of the vertex domain system for graph signals in (19), given by $H(\lambda_k) = h_0 + h_1 \lambda_k^1 + \dots + h_{M-1} \lambda_k^{M-1}$, should be equal to the desired transfer function, $G(\lambda_k)$, for each spectral index, k , leads to a system of linear equations

$$\begin{aligned} h_0 + h_1 \lambda_1^1 + \dots + h_{M-1} \lambda_1^{M-1} &= G(\lambda_1) \\ h_0 + h_1 \lambda_2^1 + \dots + h_{M-1} \lambda_2^{M-1} &= G(\lambda_2) \\ &\vdots \\ h_0 + h_1 \lambda_N^1 + \dots + h_{M-1} \lambda_N^{M-1} &= G(\lambda_N) \end{aligned} \quad (21)$$

of which the matrix form is given by

$$\mathbf{V}_\lambda \mathbf{h} = \mathbf{g}, \quad (22)$$

where \mathbf{V}_λ is a Vandermonde matrix formed of the eigenvalues, λ_k , while $\mathbf{h} = [h_0, h_1, \dots, h_{M-1}]^T$ is the vector of system coefficients that we wish to estimate, and

Comments on the Graph Filter in (22)

Consider the following cases.

- 1) All the eigenvalues of \mathbf{S} are distinct:
 - a) For $M=N$, the solution is unique.
 - b) For $M < N$ (overdetermined system), the least squares solution is obtained.
- 2) Some of the eigenvalues are of a degree higher than one, the system reduces to $N_m < N$ linear equations.
 - a) For $N_m < M \leq N$ (underdetermined system), $(M - N_m)$ filter coefficients are free variables, and an infinite number of *equivalent filters* is obtained.
 - b) For $M = N_m$, the solution is unique.
 - c) For $M < N_m$ (overdetermined system), the least squares solution is obtained.
- 3) Any filter of an order $M > N_m$ has a *unique equivalent filter* whose order is at most N_m . Such equivalence can be obtained by setting the free variables to zero, $h_i = 0$ for $i = N_m + 1, \dots, M - 1$.

Graph Topology Based on Signal Similarity: An Image Processing Example

The graph weights in our temperature field example are defined based on the geometric distance of vertices (sensing points). However, in some applications signal values themselves may be used as an indicator of signal similarity, as is the case with image processing, where this is achieved in combination with the pixel/vertex distances. For the image intensity values at pixels indexed by n and m , denoted by $x(n)$ and $x(m)$, a simple difference of intensities

$$\text{intensity distance}(n, m) = s_{nm} = |x(n) - x(m)|,$$

may be used in an exponential kernel to define the corresponding edge weights as

$$W_{nm} = e^{-|x(n) - x(m)|^2 / \tau^2} \text{ for } r_{nm} \leq \kappa,$$

and $W_{nm} = 0$ for $r_{nm} > \kappa$, where r_{nm} is a geometric distance of the considered pixels vertices, and κ is a threshold.

We next present an example of this kind of edge weighting applied to a simple graph image filtering problem.

Example

Consider the problem of denoising a 50-pixel-square, 8-bit grayscale,

image (see Figure S3). The vertices of the graph are the pixels. The edge weights for the graph representation of this noisy image were calculated with $\kappa = \sqrt{2}$ and $\tau = 20$. This value of κ means that each vertex is connected with eight neighboring vertices (including diagonal ones) with the weights, W_{nm} , defined by the exponential kernel. Low-pass filtering was performed on the corresponding image graph using iterative filtering (Taubin's algorithm) over 200 iterations, with $\alpha = 0.15$ and $\beta = 0.1$.

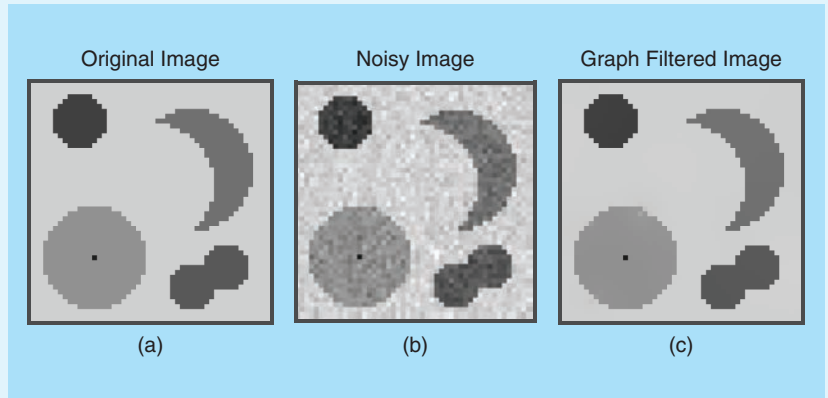


FIGURE S3. Graph domain image denoising. The (a) original image, (b) noise-corrupted image, and (c) image filtered using Taubin's algorithm (see "Smoothness and Filtering on a Graph").

$$\mathbf{g} = [G(\lambda_1), G(\lambda_2), \dots, G(\lambda_N)]^T \\ = \text{diag}(G(\Lambda)).$$

In practice, the system order M , or equivalently the order of the graph filter, is typically significantly lower than the number of equations, N , in (21). For such an overdetermined case, the least-squares approximation of \mathbf{h} is obtained by minimizing the squared error, $e^2 = \|\mathbf{V}_\lambda \mathbf{h} - \mathbf{g}\|^2$. As in standard least-squares, the solution is obtained by a direct minimization, $\partial e^2 / \partial \mathbf{h}^T = \mathbf{0}$, to yield the coefficient estimates, $\hat{\mathbf{h}}$, in the form

$$\hat{\mathbf{h}} = (\mathbf{V}_\lambda^T \mathbf{V}_\lambda)^{-1} \mathbf{V}_\lambda^T \mathbf{g} = \text{pinv}(\mathbf{V}_\lambda) \mathbf{g}. \quad (23)$$

The so obtained solution, $\hat{\mathbf{h}}$, therefore represents the mean-square error minimizer for $\mathbf{V}_\lambda \mathbf{h} = \mathbf{g}$. Notice that this solution may not satisfy $\mathbf{V}_\lambda \hat{\mathbf{h}} = \mathbf{g}$, in which case the coefficients $\hat{\mathbf{g}}$ (its spectrum $\hat{G}(\Lambda)$) may be used, that is

$$\mathbf{V}_\lambda \hat{\mathbf{h}} = \hat{\mathbf{g}}.$$

Such a solution, in general, differs from the desired system coefficients \mathbf{g} (its spectrum $G(\Lambda)$).

The desired transfer function $G(\lambda)$ can be approximated with a polynomial of degree $(M-1)$, for example, using the Chebyshev polynomial series, $P_{M-1}(\lambda)$, within the specified interval of λ , as continuous variable. Signal filtering is then performed in the vertex domain using relation (11) with a matrix polynomial $H(\mathbf{S}) = P_{M-1}(\mathbf{S})$.

Example

Consider the graph signal from Figure 3 and the graph Laplacian employed as the shift operator. The task is to design a graph filter whose frequency response is $g(\lambda_k) = \exp(-\lambda_k)$, to filter the graph signal using this spectral domain graph filter. For $M = 4$, the corresponding system coefficients can be found to be $h_0 = 0.9606$, $h_1 = -0.7453$,

$h_2 = 0.1936$, and $h_3 = -0.0162$. Upon filtering the graph signal using the so-defined graph transfer function, and its vertex domain form (17), the obtained output signal-to-noise ratio was $\text{SNR} = 21.74$ dB, that is, a 7.54 dB improvement over the original signal-to-noise ratio of $\text{SNR}_0 = 14.2$ dB. More detail on the solution of the system in (21) and (22) is provided in "Comments on the Graph Filter in (22)."

Optimal denoising

Consider a measurement, as in the temperature measurement scenario in Figure 1, that is composed of a slow-varying desired signal, \mathbf{s} , and a superimposed fast-changing disturbance, ε , to give

$$\mathbf{x} = \mathbf{s} + \varepsilon.$$

The aim is to design a graph filter for disturbance suppression (denoising), the output of which is denoted by \mathbf{y} [11].

The optimal denoising task can then be defined through a minimization of the cost function

$$J(\mathbf{y}|\mathbf{x}) = \frac{1}{2} \|\mathbf{y} - \mathbf{x}\|_2^2 + \alpha \mathbf{y}^T \mathbf{L} \mathbf{y}, \quad (24)$$

where the minimization of the first term, $1/2 \|\mathbf{y} - \mathbf{x}\|_2^2$, enforces the output signal, \mathbf{y} , to be as close as possible, in terms of the minimum residual disturbance power, to the available observations, \mathbf{x} . As shown in “Smoothness and Filtering on a Graph,” the second term, $\mathbf{y}^T \mathbf{L} \mathbf{y}$, represents a measure of smoothness of the graph filter output, \mathbf{y} , while the parameter α models a balance between the closeness of the output, \mathbf{y} , to the observed data, \mathbf{x} , and the smoothness-constrained output estimate \mathbf{y} . While the problem in (24) could also be expressed through a constrained Lagrangian optimization, we here focus more on the graph-theoretic issues and hence adopt a simpler option whereby the mixing parameter α is chosen empirically.

The solution to the minimization problem in (24) follows from

$$\frac{\partial J(\mathbf{y}|\mathbf{x})}{\partial \mathbf{y}^T} = \mathbf{y} - \mathbf{x} + 2\alpha \mathbf{L} \mathbf{y} = \mathbf{0}$$

and results in a smoothing optimal denoiser in the form

$$\mathbf{y} = (\mathbf{I} + 2\alpha \mathbf{L})^{-1} \mathbf{x}.$$

The Laplacian spectral domain form of this relation then becomes

$$\mathbf{Y} = (\mathbf{I} + 2\alpha \mathbf{\Lambda})^{-1} \mathbf{X},$$

with the corresponding graph filter transfer function given by

$$H(\lambda_k) = \frac{1}{1 + 2\alpha \lambda_k}.$$

Observe that for a small α , $H(\lambda_k) \approx 1$ and $\mathbf{y} \approx \mathbf{x}$, while for a large α , $H(\lambda_k) \approx \delta(k)$ and $\mathbf{y} \approx \text{const.}$, which enforces \mathbf{y} to be maximally smooth (a constant, without any variation). Using $\alpha = 4$, the obtained output signal-to-noise ratio for the graph signal from Figure 3 was $\text{SNR} = 26$ dB,

an 11.8 dB improvement over the original $\text{SNR}_0 = 14.2$ dB.

Remark 11

There are many cases when the graph topology is unknown, so that the graph structure, that is, the graph edges and their weights, is also unknown. To this end, we may employ a class of methods for graph topology learning, which are based on the minimization of the cost function in (24) with respect to both a chosen graph connectivity matrix and the output signal, \mathbf{y} , with additional (commonly sparsity) constraints imposed on the elements of the considered connectivity matrix.

What we have learned

Natural signals (speech, biomedical, video) often reside over irregular domains and are, unlike the standard signals in, for example, communications, not adequately processed using standard harmonic analyses. While data analytics on graphs as irregular domains are heavily dependent on advances in DSP, neither the electrical and electronics engineering graduates worldwide nor practical data analysts are yet best prepared to employ graph algorithms in their future jobs. Our aim has been to fill this void by providing an example-driven platform to introduce graphs, signals on graphs, and their properties through a well-understood multisensor measurement scenario and the graph notions of transfer function, Fourier transform, and digital filtering.

We have illuminated that while both a graph with N vertices and a classical discrete time signal with N samples can be viewed as N -dimensional vectors, structured graphs represent much richer irregular domains that convey information about both the signal itself and its generation and propagation mechanisms. This allows us to employ intuition and physical know-how from Euclidean domains to revisit basic dimensionality-reduction operations, such as coarse graining of graphs (cf. standard downsampling). In addition, in the vertex domain a number of different distances (shortest-path, resistance, diffusion) have useful properties that can be employed to maintain

data integrity throughout the processing, storage, communication, and analysis stages, as the vertex connectivities and edge weights are either dictated by the physics of the problem at hand or are inferred from the data. This particularly facilitates maintaining control and intuition over distributed operations throughout the processing chain.

It is our hope that this lecture note will help to demystify graph signal processing for students and educators, together with empowering practitioners with enhanced intuition in graph-theoretic design and optimization. This material may also serve as a vehicle to seamlessly merge curricula in electrical engineering and computing. The generic and physically meaningful nature of this example-driven lecture note is also likely to promote intellectual curiosity and serve as a platform to explore the numerous opportunities in manifold applications in our ever-growing interconnected world, facilitated by the Internet of Things.

Acknowledgments

We give our sincere thanks to the anonymous reviewers for their constructive and insightful comments. We are privileged to have had the help and advice of one of the pioneers in graph theory, Prof. Nicos Christofides. We are grateful for his time, his incisive comments, and valuable advice. The help from Bruno Scalzo Dees, Alexandros Haliassos, and Shengxi Li from Imperial College London and Miloš Brajović from the University of Montenegro is also greatly appreciated. Last but not least, we would also like to express our sincere gratitude to the students in our respective postgraduate courses for their feedback on the material taught based on this lecture note.

Authors

Ljubiša Stanković (ljubisa@ucg.ac.me) is a professor at the University of Montenegro. His research interests include time–frequency analysis, compressive sensing, and graph signal processing. He is a vice president of the National Academy of Sciences and Arts of Montenegro and is a recipient of the 2017 European Association for

Signal Processing Best Journal Paper Award. He is a Fellow of the IEEE.

Danilo P. Mandić (d.mandic@imperial.ac.uk) is a professor of signal processing at Imperial College London, United Kingdom. He has a keen interest in signal processing education, is a member of the IEEE Signal Processing Society Education Technical Committee, and his contributions were recognized with the President's Award for Excellence in Postgraduate Supervision at Imperial College in 2014. He is a Fellow of the IEEE.

Miloš Daković (milos@ucg.ac.me) is professor at the University of Montenegro. His research interests include graph signal processing, compressive sensing, and time–frequency analysis. He is a Member of the IEEE.

Ilya Kisil (i.kisil15@imperial.ac.uk) is a Ph.D. candidate at Imperial College London, United Kingdom. His research interests include tensor decompositions, big data, efficient software for large-scale problems, and graph signal processing.

Ervin Sejdić (esejdic@ieee.org) is an associate professor at the University of Pittsburgh, Pennsylvania. His research interests include biomedical signal processing, rehabilitation engineering, and neuroscience. He received the U.S. Presidential Early Career Award for Scientists and Engineers in 2016. He is a Senior Member of the IEEE.

Anthony G. Constantinides (a.constantinides@imperial.ac.uk) is emeritus professor of signal processing in the Department of Electrical and Electronic Engineering at Imperial College London, United Kingdom. He is a Life Fellow of the IEEE.

References

- [1] N. Christofides, *Graph theory: An algorithmic approach*. New York: Academic, 1975.
- [2] F. Afrati and A. G. Constantinides, "The use of graph theory in binary block code construction," in *Proc. Int. Conf. on Digital Signal Processing*, Florence, Italy, 1978, pp. 228–233.
- [3] O. J. Morris, M. de J. Lee, and A. G. Constantinides, "Graph theory for image analysis: An approach based on the shortest spanning tree," *IEE Proc. F Commun.*, *Radar Signal Processing*, vol. 133, no. 2, pp. 146–152, 1986. doi: 10.1049/fp-f-1.1986.0025.
- [4] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, "Discrete signal processing on graphs: Sampling theory," *IEEE Trans. Signal Process.*, vol. 63, no. 24, pp. 6510–6523, Dec. 2015. doi: 10.1109/TSP.2015.2469645.
- [5] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1644–1656, Apr. 2013. doi: 10.1109/TSP.2013.2238935.
- [6] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Frequency analysis," *IEEE Trans. Signal Process.*, vol. 62, no. 12, pp. 3042–3054, June 2014. doi: 10.1109/TSP.2014.2321121.
- [7] G. Cheung, E. Magli, Y. Tanaka, and M. K. Ng, "Graph spectral image processing," *Proc. IEEE*, vol. 106, no. 5, pp. 907–930, May 2018. doi: 10.1109/JPROC.2018.2799702.
- [8] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, May 2018. doi: 10.1109/JPROC.2018.2820126.
- [9] S. Saito, H. Suzuki, and D. P. Mandić, "Hypergraph p-Laplacian: A differential geometry view," in *Proc. 32nd AAAI Conf. on Artificial Intelligence (AAAI-18)*, 2018, pp. 3984–3991.
- [10] L. Stanković and E. Sejdić, *Vertex-Frequency Analysis of Graph Signals*. Cham, Switzerland: Springer Nature, 2019.
- [11] S. Segarra, A. G. Marques, and A. Ribeiro, "Optimal graph-filter design and applications to distributed linear network operators," *IEEE Trans. Signal Process.*, vol. 65, no. 15, pp. 4117–4131, 2017. doi: 10.1109/TSP.2017.2703660.

Víctor Elvira, Luca Martino, Mónica F. Bugallo, and Petar M. Djurić

Elucidating the Auxiliary Particle Filter via Multiple Importance Sampling

Sequential Monte Carlo methods, also known as *particle filtering*, have seen an explosion of development both in theory and applications. The publication of [1] sparked huge interest in the area of sequential signal processing, particularly in sequential filtering. Ever since,

the number of publications in which particle filtering plays a prominent role has continued to grow. An early reference of development is [2] and later tutorials include [3]–[9]. With particle filtering, we estimate probability density functions (pdfs) of interest by probability mass functions, whose masses are placed at randomly chosen locations (particles) and whose weights are assigned to the particles.

The particle filter (PF) proposed in [1] is often called the *bootstrap PF (BPF)*, and although it is not optimal, it is the most often used filter by practitioners. A filter that became also popular is known as the *auxiliary PF (APF)* and was proposed in [10]. With the APF, the objective is to generate better particles at each time step compared to those generated with the BPF, thereby improving filtering accuracy.

In this article, we derive the APF from a new perspective, one based on interpreting the APF from the multiple importance sampling (MIS) paradigm. The derivation also shows its relationship with the BPF.

Relevance

State-space models are ubiquitously used in many fields of science and engineering for modeling complex dynamical systems. Once the models are formulated, the usual goal is to estimate the state of the model, which evolves over time, from observations that are sequentially acquired and functions of the state. With the Bayesian approach, the estimation is carried out through the posterior distribution of the state, which can be obtained analytically for linear and Gaussian models (where the Kalman filter is applied [11]). By contrast, PFs allow for the approximation of the posterior for virtually any state-space model, even models used for the most complex systems. The BPF is a simple implementation of the PF, but advanced filters such as the APF are necessary to boost the performance in complex and higher-dimensional scenarios. Although an APF often obtains better results than does a BPF, in some settings, the performance is similar or even worse.

The novel derivation presented in this article will help the reader to 1) gain new insights about APFs and 2) better understand when it is best to use one. Note that APFs have been used in many real-world problems, such as mathematical finance [12], tracking applications [13], sensor networks [14], or electronics [15], among many others.

Prerequisites

This article is designed for researchers and students who have a basic knowledge of particle filtering and IS. It aims to provide a new interpretation of the popular APF and a better understanding of this specific type of PF. The reader is expected to be familiar with basic notions of PFs [4], including the basic derivations of BPFs [16], [17], and some familiarity with APFs is preferable [10]. Finally, the novel perspective presented in this article will be better understood with some basic knowledge of the IS methodology [18].

Problem statement

Bayesian filtering in dynamical models

We consider the following Markovian state-space model in discrete time ($t \in \mathbb{N}^+$):

$$\mathbf{x}_0 \sim p(\mathbf{x}_0), \quad (1)$$

$$\mathbf{x}_t \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}), \quad (2)$$

$$\mathbf{y}_t \sim p(\mathbf{y}_t | \mathbf{x}_t), \quad (3)$$

where $\mathbf{x}_t \in \mathbb{R}^{d_x}$ represents the hidden (and random) system state at time instant t ; $p(\mathbf{x}_0)$ is the a priori pdf of the state at t_0 ; $p(\mathbf{x}_t | \mathbf{x}_{t-1})$ denotes the transition kernel (i.e., the conditional density of the state \mathbf{x}_t given \mathbf{x}_{t-1}); $\mathbf{y}_t \in \mathbb{R}^{d_y}$ is the observation vector at time instant t , and is assumed to be conditionally independent of all other observations given the state \mathbf{x}_t ; and $p(\mathbf{y}_t | \mathbf{x}_t)$ is the conditional pdf of \mathbf{y}_t given \mathbf{x}_t , (i.e., the observation kernel) which is often referred to as the *likelihood* of \mathbf{x}_t . The model described by (1)–(3) includes a broad class of systems, both linear and nonlinear, with Gaussian or non-Gaussian perturbations.

The filtering problem consists of the probabilistic estimation of the hidden state \mathbf{x}_t conditioned on all the observations available up to time instant t . For simplicity in the notation, we denote this set of observations $\mathbf{y}_{1:t} = \{\mathbf{y}_\tau\}_{\tau=1}^t$. Hence, the goal is the computation of the filtering pdf $p(\mathbf{x}_t | \mathbf{y}_{1:t})$. When the observations are received sequentially, the additional goal is to process the observations recursively. To this end, the filtering task requires two steps at each time instant t :

- 1) *Propagation step*: The predictive pdf of the state, $p(\mathbf{x}_t | \mathbf{y}_{1:t-1})$, is computed as

$$p(\mathbf{x}_t | \mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_t | \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} | \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}. \quad (4)$$

- 2) *Update step*: According to Bayes' theorem and from (3) and (4), the filtering distribution is obtained by

$$p(\mathbf{x}_t | \mathbf{y}_{1:t}) \propto p(\mathbf{y}_t | \mathbf{x}_t) \int p(\mathbf{x}_t | \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} | \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}. \quad (5)$$

Unfortunately, these two steps require solving intractable integrals for most models of interest, and as a result, the filtering distribution cannot be exactly obtained. Thus, one has to search for approximate solutions, and to that end, Monte Carlo-based methods offer a rich trove of possibilities.

Particle filtering

Particle filtering is a Monte Carlo technique where the distributions are approximated by sets of weighted random samples in a sequential manner. For simplicity, here we assume that all algorithms always have M particles at each time instant t , although strategies for adapting the number of particles with time have been recently proposed [19].

BPF

The BPF is arguably the most well-known PF algorithm. It is often called a *sequential importance resampling filter* because it implements a sequential IS step followed by a resampling step. The outline of the BPF is described in Algorithm 1. Note that the weights of (A2) are normalized in the sense that $\sum_{m=1}^M w_t^{(m)} = 1$.

The APF

The APF aims to improve the “quality” of the samples that are generated at each time instant. Here, *quality* means how

Algorithm 1: The bootstrap particle filter.

- 1) *Initialization*: At time instant $t = 0$, draw M independent and identically distributed samples, $\mathbf{x}_0^{(m)}$, $m = 1, \dots, M$, from the distribution $p(\mathbf{x}_0)$.
- 2) *Recursive step*: Let $\{\mathbf{x}_{t-1}^{(m)}\}_{m=1}^M$ be the particles (samples) resampled at time instant $t-1$. At time instant t , proceed with the following steps:
 - *Propagation step*: Propagate the particles as

$$\mathbf{x}_t^{(m)} \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(m)}), \quad m = 1, \dots, M. \quad (A1)$$
 - *Update step*: Compute the normalized weights as

$$w_t^{(m)} \propto p(\mathbf{y}_t | \mathbf{x}_t^{(m)}), \quad m = 1, \dots, M. \quad (A2)$$
 - *Resampling step*: Resample M times from the approximation of the filtering distribution as

$$\tilde{\mathbf{x}}_t^{(m)} \sim \sum_{j=1}^M w_t^{(j)} \delta(\mathbf{x}_t - \mathbf{x}_t^{(j)}). \quad (A3)$$

representative the drawn samples are of the target pdf. In principle, they are more representative if they come from the parts of \mathbb{R}^{d_x} that contain higher probability masses as measured by the target pdf. To achieve this improvement, unlike the BPF, the APF uses the new observation \mathbf{y}_t in the prediction step, which is implemented by way of a delayed resampling (avoided at the end of the previous time instant).

The outline of the APF is described in Algorithm 2. First, (A4) is used to generate preparticles (particle projections) based on the expected value given the particles from the previous step. Note that the values $\tilde{\mathbf{x}}_t^{(m)}$, $m = 1, 2, \dots, M$, represent the centers of the kernels that compose the usual PF approximation of the predictive distribution of \mathbf{x}_t :

$$p(\mathbf{x}_t | \mathbf{y}_{1:t-1}) \approx p^M(\mathbf{x}_t | \mathbf{y}_{1:t-1}) = \sum_{m=1}^M w_{t-1}^{(m)} p(\mathbf{x}_t | \tilde{\mathbf{x}}_{t-1}^{(m)}). \quad (6)$$

Then, (A5) is applied to compute the weights of $\tilde{\mathbf{x}}_t^{(m)}$ using the current observation \mathbf{y}_t . In the literature, these weights are called *preweights*. The preweights are employed to perform a delayed resampling in the delayed resampling step. Hence, the particles are replicated taking into account not only the previous observation (as in BPF) but also using the information of the current observation. The final set of particles at time t , $\mathbf{x}_t^{(m)}$, comes from the propagation described by (A6). Finally, the particles receive a weight computed according to (A7). Consider that this weight is proportional to the likelihood at time instant t , but inversely proportional to $p(\mathbf{y}_t | \tilde{\mathbf{x}}_t^{(m)})$. One can interpret that the m th particle is replicated proportionally to $w_{t-1}^{(m)} p(\mathbf{y}_t | \tilde{\mathbf{x}}_t^{(m)})$ instead of $w_{t-1}^{(m)}$ (as in the BPF), and that the final IS weight “discounts” this augmentation of the preweight by dividing the likelihood of the particle by the factor $p(\mathbf{y}_t | \tilde{\mathbf{x}}_t^{(m)})$. Note that as a result, the APF approximates the filtering distribution at time instant t , $p(\mathbf{x}_t | \mathbf{y}_{1:t})$, with the random measure $p^M(\mathbf{x}_t | \mathbf{y}_{1:t}) = \sum_{m=1}^M w_t^{(m)} \delta(\mathbf{x}_t - \mathbf{x}_t^{(m)})$.

A unifying framework

The usual description of the APF presented in Algorithm 2 does not allow

Algorithm 2: The auxiliary particle filter.

- 1) *Initialization*: At time instant $t = 0$, draw M independent and identically distributed samples, $\mathbf{x}_0^{(m)}$, $m = 1, \dots, M$, from the distribution $p(\mathbf{x}_0)$.
- 2) *Recursive step*: Let $\{\mathbf{x}_{t-1}^{(m)}, w_{t-1}^{(m)}\}_{m=1}^M$ be the set of weighted particles (samples) generated at time instant $t-1$. At time instant t , proceed with the following steps:

- *Prewrite computation step*:
 - Compute the mean of the probability density function $p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(m)})$ as

$$\tilde{\mathbf{x}}_t^{(m)} = \mathbb{E}_{p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(m)})}[\mathbf{x}_t], \quad m = 1, \dots, M. \quad (A4)$$

- Compute the normalized preweights of each kernel in the mixture as

$$\lambda_t^{(m)} \propto p(\mathbf{y}_t | \tilde{\mathbf{x}}_t^{(m)}) w_{t-1}^{(m)}, \quad m = 1, \dots, M. \quad (A5)$$

- *Delayed resampling step*: Sample the indexes $i^{(m)}$, $m = 1, \dots, M$, with probability mass function given by $\mathbb{P}(i^{(m)} = j) = \lambda_t^{(j)}$, $j \in \{1, \dots, M\}$.
- *Propagation step*: Simulate

$$\mathbf{x}_t^{(m)} \sim p(\mathbf{x}_t | \tilde{\mathbf{x}}_{t-1}^{(i^{(m)})}), \quad m = 1, \dots, M. \quad (A6)$$

- *Update step*: Compute the normalized weights as

$$w_t^{(m)} \propto \frac{p(\mathbf{y}_t | \mathbf{x}_t^{(m)})}{p(\mathbf{y}_t | \tilde{\mathbf{x}}_t^{(i^{(m)})})}, \quad m = 1, \dots, M. \quad (A7)$$

for a clear comparison with the BPF, and in particular, neither the full justification of the derivation of the preweights of (A5), nor the demonstration of the validity of the weights of (A7). In the following section, we develop a unifying framework based on MIS that eases the derivation of several existing PF algorithms. The MIS perspective also makes easier the understanding of some of the challenges of the APF and the proposing of novel algorithms.

Common framework

The novel unifying framework is presented in Algorithm 3. Interestingly, it avoids the explicit use of the resampling step, which usually hinders its connection with IS; however, bear in mind that sampling from a proposal mixture is equivalent to a resampling step followed by a propagation step. Therefore, the generic algorithm is presented in a simpler adaptation-sampling-weighting manner. The initialization of the algorithm is performed as usual, and the recursive step also uses as a basis the set of available weighted particles $\{\mathbf{x}_{t-1}^{(m)}, w_{t-1}^{(m)}\}_{m=1}^M$ from time $t-1$. In (A8), a proposal, $\psi(\mathbf{x}_t)$, is selected/adapted. Note that this proposal is composed of the transition ker-

nels $\{p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(m)})\}_{m=1}^M$, whose positions depend on each of the M particles at $t-1$. The dissimilar algorithms differ in the selection of associated coefficients $\{\lambda_t^{(m)}\}_{m=1}^M$ (see the “MIS Perspective” section). In (A12), the M new particles are independent and identically distributed (i.i.d.) simulated from ψ . Finally, the normalized IS weights are proportionally computed, as in (A16). Note that this weight is supported by simple IS arguments. The numerator is proportional to the filtering distribution $p(\mathbf{x}_t | \mathbf{y}_{1:t-1}) \propto p(\mathbf{y}_t | \mathbf{x}_t) p(\mathbf{x}_t | \mathbf{y}_{1:t-1})$, where $p(\mathbf{x}_t | \mathbf{y}_{1:t-1})$ is substituted by the standard particle approximation of the predictive distribution given by (6). The denominator is the proposal $\psi(\mathbf{x}_t)$. Both the numerator and denominator are evaluated at each particle (i.e., the usual targeted-divided-by-proposal IS weight).

MIS perspective

In Algorithm 3, we described a generic PF from the MIS perspective where the traditional propagation-update-resampling steps are now replaced by adaptation-sampling-weighting steps. The sampling in the generic PF framework is performed by (A12), where M samples are simulated from the mixture proposal.

Algorithm 3: The multiple importance sampling interpretation of particle filtering.

- 1) *Initialization*: At time $t = 0$, draw M independent and identically distributed samples, $\mathbf{x}_0^{(m)}$, $m = 1, \dots, M$, from the distribution $p(\mathbf{x}_0)$, and set $\lambda_1^{(m)} = 1/M$.
- 2) *Recursive step*: Let $\{\mathbf{x}_{t-1}^{(m)}, w_{t-1}^{(m)}\}_{m=1}^M$ be the set of weighted particles (samples) generated at time $t-1$. At time t , proceed with the following steps:
 - *Proposal adaptation/selection*: Select the multiple importance sampling proposal of the form

$$\psi_t(\mathbf{x}_t) = \sum_{m=1}^M \lambda_t^{(m)} p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(m)}), \quad (\text{A8})$$

where $\{p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(m)})\}_{m=1}^M$ are the transition kernels centered at each of the M particles at $t-1$, and $\{\lambda_t^{(m)}\}_{m=1}^M$ are the associated coefficients computed by

$$\lambda_t^{(m)} = w_{t-1}^{(m)}, \quad m = 1, \dots, M, \quad \text{if the applied filter is the bootstrap particle filter,} \quad (\text{A9})$$

$$\lambda_t^{(m)} \propto p(\mathbf{y}_t | \mathbf{x}_t^{(m)}) w_{t-1}^{(m)}, \quad m = 1, \dots, M, \quad \text{if the applied filter is the auxiliary particle filter,} \quad (\text{A10})$$

where

$$\bar{\mathbf{x}}_t^{(m)} = \mathbb{E}_{p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(m)})}[\mathbf{x}_t], \quad m = 1, \dots, M. \quad (\text{A11})$$

are the means of the kernels.

- *Sampling*: Draw samples according to

$$\mathbf{x}_t^{(m)} \sim \psi_t(\mathbf{x}_t), \quad m = 1, \dots, M. \quad (\text{A12})$$

- *Weighting*: Compute the normalized importance sampling weights by

$$w_t^{(m)} \propto \frac{p(\mathbf{x}_t^{(m)} | \mathbf{y}_{1:t})}{\psi_t(\mathbf{x}_t^{(m)})} \quad (\text{A13})$$

$$= \frac{p(\mathbf{y}_t | \mathbf{x}_t^{(m)}) p(\mathbf{x}_t^{(m)} | \mathbf{y}_{1:t-1})}{\psi_t(\mathbf{x}_t^{(m)})} \quad (\text{A14})$$

$$\approx \frac{p(\mathbf{y}_t | \mathbf{x}_t^{(m)}) \sum_{j=1}^M w_{t-1}^{(j)} p(\mathbf{x}_t^{(m)} | \mathbf{x}_{t-1}^{(j)})}{\psi_t(\mathbf{x}_t^{(m)})} \quad (\text{A15})$$

$$= \frac{p(\mathbf{y}_t | \mathbf{x}_t^{(m)}) \sum_{j=1}^M w_{t-1}^{(j)} p(\mathbf{x}_t^{(m)} | \mathbf{x}_{t-1}^{(j)})}{\sum_{j=1}^M \lambda_t^{(j)} p(\mathbf{x}_t^{(m)} | \mathbf{x}_{t-1}^{(j)})} \quad (\text{A16})$$

$$\approx \frac{p(\mathbf{y}_t | \mathbf{x}_t^{(m)}) w_{t-1}^{(m)}}{\lambda_t^{(m)}}, \quad m = 1, \dots, M. \quad (\text{A17})$$

Let us now focus on the adaptation/selection of the proposal and its impact on the weighting step of (A16). It is well known that in MIS, when we draw i.i.d. samples from a mixture, the performance improves when the discrepancy between the proposal and the target density decreases [18]. In other words, it is important that the numerator of (A15), $p(\mathbf{x}_t | \mathbf{y}_{1:t})$ and the denominator, $\psi(\mathbf{x}_t)$, are as similar as possible. Note that, in the IS weight, the numerator in (A16) is the product of the likelihood

and a mixture of weighted kernels, while the denominator is another mixture with the same kernels. The fact that both mixtures have the same kernels with (potentially) different coefficients is key in the derivation of the different algorithms as well as in the understanding of their performance under different models.

Table 1 summarizes the choice of preweights and weights of the BPF, APF, and the improved APF (IAPF) [20], a recent algorithm that also fits

in the MIS framework previously mentioned. In the following section, we provide details of the new interpretations for each algorithm.

BPF from the MIS perspective

Let us interpret the BPF of Algorithm 1 from the generic Algorithm 3. First, set $\lambda_t^{(j)} = w_{t-1}^{(j)}$ in (A8). Sampling M times from this mixture $\psi_t(\mathbf{x}_t) = \sum_{m=1}^M w_{t-1}^{(m)} p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(m)})$ is equivalent to the resampling step at time instant $t-1$ from (A2), followed by the propagation of particles in (A1). If we plug $\psi_t(\mathbf{x}_t)$ in the denominator of (A16), because $\psi_t(\mathbf{x}_t) = p(\mathbf{x}_t^{(m)} | \mathbf{y}_{1:t-1})$, it cancels out the right-hand side of the numerator and only the likelihood in the numerator remains. This allows us to recover the traditional BPF weight $w_t^{(m)} \propto p(\mathbf{y}_t | \mathbf{x}_t^{(m)})$ of (A2), which is simply the likelihood evaluated at the associated particle [compare (23)]. We point out that, for the BPF, the approximation in (A17) becomes an equality because the summations in the numerator and denominator of (A16) are identical, and before we implement the approximation in (A17), they cancel each other.

In BPF, the strategy for the choice of the coefficients $\lambda_t^{(m)} = w_{t-1}^{(m)}$ in the proposal mixture is to match the mixture of the numerator of (A16), where the likelihood in the numerator is a mismatch factor between the numerator and denominator. This explains the challenges of BPF when there are very informative observations: The likelihood can severely amplify some kernels with respect to (w.r.t.) others, thus increasing the mismatch between the target and the proposal distributions.

Figure 1 shows the traditional and MIS perspectives of the BPF, respectively. In Figure 1(a), we display the traditional propagation, weighting, and resampling steps, which are repeated at each time step. In Figure 1(b), we display the MIS interpretation, with the proposal selection, the sampling from a mixture, and the weighting. From the MIS perspective, the sampling step from the mixture of proposal [ψ_t in (A8)] is equivalent to resampling, followed by propagation of kernels from the traditional perspective. In the BPF,

there is no proposal/selection adaptation because the mixture proposal $\psi_t(\mathbf{x}_t)$ depends exclusively on the kernels and the IS weights of the previous time step.

The APF from the MIS perspective

From the MIS perspective, the choice of the APF seems more adequate than that of the BPF. Although the APF still contains the same kernels in the proposal mixture, its coefficients $\lambda_t^{(m)} \propto p(\mathbf{y}_t | \tilde{\mathbf{x}}_t^{(m)}) w_{t-1}^{(m)}$ are those of the BPF but are amplified by the factor $p(\mathbf{y}_t | \tilde{\mathbf{x}}_t^{(m)})$. This factor is simply the likelihood evaluated at the mean of the m th kernel. For instance, for additive Gaussian perturbations in the state model, the mixture proposal of the APF is the same as that of the mixture proposal of the BPF, where each kernel has been amplified by the value of the function $p(\mathbf{y}_t | \mathbf{x}_t)$ at its center. Interestingly, this approach now tries to minimize the mismatch of $\psi(\mathbf{x}_t)$ with the whole numerator in (A16), and not only with a part of it, i.e., with the factor that corresponds to the predictive pdf $p(\mathbf{x}_t | \mathbf{y}_{1:t-1})$. In many state-space models of interest, the APF has improved performance due to this better-matching mixture proposal. Note that this approximation of the target is especially good when the kernels are “far apart” (more precisely, when their distance is high w.r.t. their width).

Within the MIS perspective, we also derive the weights of the APF from (A16) under the assumption of well-separated kernels. Let us consider that the m th particle $\mathbf{x}_t^{(m)}$ has been simulated from the kernel $i^{(m)}$. If the other kernels, e.g., $p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(j)})$ with $j = 1, \dots, M$ and $j \neq i^{(m)}$, take small values when evaluated at $\tilde{\mathbf{x}}_t^{(i^{(m)})}$ (which is equivalent to our previous assumption of the kernels being separated enough), one can approximate the weight as

$$w_t^{(m)} \propto \frac{p(\mathbf{x}_t^{(m)} | \mathbf{y}_{1:t})}{\psi_t(\mathbf{x}_t^{(m)})} \quad (7)$$

$$\approx \frac{p(\mathbf{y}_t | \mathbf{x}_t^{(m)}) \sum_{j=1}^M w_{t-1}^{(j)} p(\mathbf{x}_t^{(m)} | \mathbf{x}_{t-1}^{(j)})}{\sum_{j=1}^M \lambda_t^{(j)} p(\mathbf{x}_t^{(m)} | \mathbf{x}_{t-1}^{(j)})} \quad (8)$$

Table 1. The coefficients $\lambda_t^{(m)}$ of the proposal mixture, and the IS weights $w_t^{(m)}$ for the BPF, APF, and IAPF. Note that in all cases, the weights must be normalized such that $\sum_{m=1}^M \lambda_t^{(m)} = 1$ and $\sum_{m=1}^M w_t^{(m)} = 1$.

| | BPF | APF | IAPF |
|-------------------|------------------------------------------------|-----------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $\lambda_t^{(m)}$ | $w_{t-1}^{(m)}$ | $\propto p(\mathbf{y}_t \tilde{\mathbf{x}}_t^{(m)}) w_{t-1}^{(m)}$ | $\propto \frac{p(\mathbf{y}_t \tilde{\mathbf{x}}_t^{(m)}) \sum_{j=1}^M w_{t-1}^{(j)} p(\mathbf{x}_t^{(m)} \mathbf{x}_{t-1}^{(j)})}{\sum_{j=1}^M p(\tilde{\mathbf{x}}_t^{(m)} \mathbf{x}_{t-1}^{(j)})}$ |
| $w_t^{(m)}$ | $\propto p(\mathbf{y}_t \mathbf{x}_t^{(m)})$ | $\propto \frac{p(\mathbf{y}_t \mathbf{x}_t^{(m)})}{p(\mathbf{y}_t \tilde{\mathbf{x}}_t^{(m)})}$ | $\propto \frac{p(\mathbf{y}_t \mathbf{x}_t^{(m)}) \sum_{j=1}^M w_{t-1}^{(j)} p(\mathbf{x}_t^{(m)} \mathbf{x}_{t-1}^{(j)})}{\sum_{j=1}^M \lambda_t^{(j)} p(\mathbf{x}_t^{(m)} \mathbf{x}_{t-1}^{(j)})}$ |

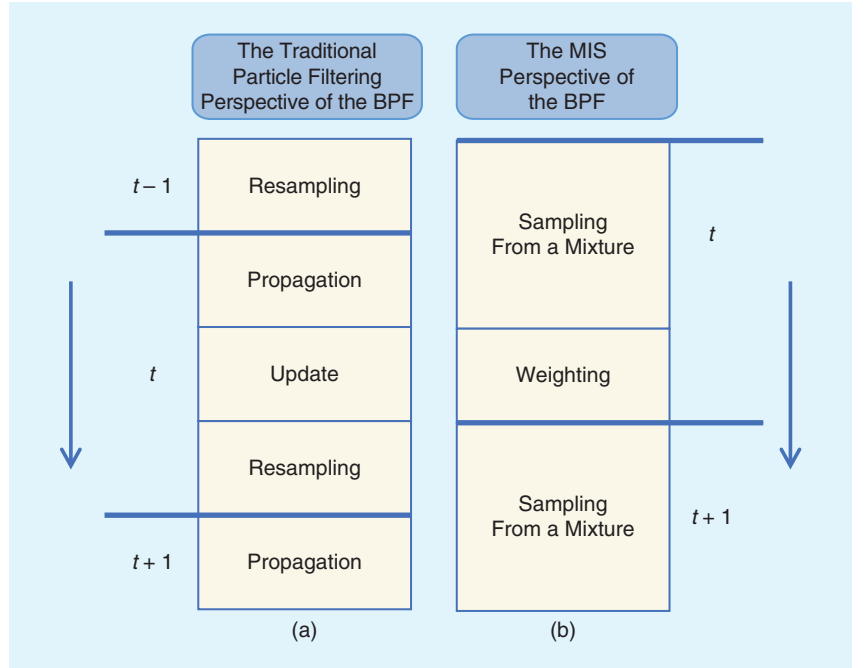


FIGURE 1. The two different perspectives for explaining the BPF. (a) The traditional perspective with the propagation, update, and resampling steps. (b) The MIS perspective with sampling from a mixture and the weighting steps.

$$\approx \frac{p(\mathbf{y}_t | \mathbf{x}_t^{(m)}) w_{t-1}^{(i^{(m)})} p(\mathbf{x}_t^{(m)} | \mathbf{x}_{t-1}^{(i^{(m)})})}{\lambda_t^{(i^{(m)})} p(\mathbf{x}_t^{(m)} | \mathbf{x}_{t-1}^{(i^{(m)})})} \quad (9)$$

$$\propto \frac{p(\mathbf{y}_t | \mathbf{x}_t^{(m)}) w_{t-1}^{(i^{(m)})} p(\mathbf{x}_t^{(m)} | \mathbf{x}_{t-1}^{(i^{(m)})})}{p(\mathbf{y}_t | \tilde{\mathbf{x}}_t^{(i^{(m)})}) w_{t-1}^{(i^{(m)})} p(\mathbf{x}_t^{(m)} | \mathbf{x}_{t-1}^{(i^{(m)})})} \quad (10)$$

$$= \frac{p(\mathbf{y}_t | \mathbf{x}_t^{(m)})}{p(\mathbf{y}_t | \tilde{\mathbf{x}}_t^{(i^{(m)})})}, \quad (11)$$

recovering the traditional APF of (A7) [or (A17)]. The first estimate in (8) comes from approximating the predictive distribution $p(\mathbf{x}_t | \mathbf{y}_{1:t})$. The second approximation in (10) assumes that the evaluation of a sample in the whole

mixture is equivalent to the evaluation in the kernel where the sample was simulated from. Hence, the latter approximation assumes that the kernels have negligible overlaps, which can be a too-strict assumption and would explain why, for some models, the APF can perform worse than the BPF.

Figure 2 shows the two interpretations of the APF described previously. In Figure 2(a), we display the traditional preweights computation, delayed resampling, propagation, and weight computation, which are repeated at each time step. In Figure 2(b), we see the MIS approach, with the proposal

selection, the (mixture) sampling, and the weighting steps. Again, the update step from the former perspective is equivalent to the weighting step from the latter perspective. In the MIS perspective, the sampling from the mixture of proposals is again equivalent to resampling, followed by propagation of kernels from the traditional perspective. The difference now (w.r.t. the BPF) is that, from the traditional perspective, there is a preweighting step for modifying the weights before the resampling step. This preweighting step is clearly equivalent to selecting the coefficients $\{\lambda_t^{(j)}\}_{j=1}^M$ of the kernels in the mixture proposal $\psi_t(\mathbf{x}_t)$ from the MIS perspective.

IAPF from the MIS perspective

An IAPF has recently been proposed to reduce the APF flaws that we listed in the previous section [20]. It takes into account the overlap of the kernels to both select the coefficients of the mixture proposal and approximate the IS weights. Algorithm 4 describes the IAPF within the novel MIS framework. First, the coefficient of the m th kernel

of the mixture in (A19) is computed by evaluating the ratio between the target distribution and the equally weighted mixture of proposals at the center of the kernel. This allows us to take into account the positions of the remaining kernels when computing the coefficient of a specific kernel (unlike in APFs). Second, the weight in (A21) is computed as it is traditionally done in IS, i.e., it is simply the ratio between the target and the whole proposal mixture evaluated at the sample.

Note that the IAPF becomes the APF when the overlap between the kernels decreases. In [20], BPF, APF, and IAPF are tested on a linear-Gaussian model, where the transition kernels are particularly wide and the observations are very informative (low observation variance). This scenario is, in principle, particularly adverse for the APF because the assumptions on the selection of the proposal do not hold and the approximation of the IS weights in (10) is poor. This explains why the IAPF can obtain a much smaller mean square error than the APF. Note, however, that the computation of the weights of IAPF

in (A21) requires two M^2 extra kernel evaluations, unlike the IS weights of the APF in (11). Similarly, the computation of the coefficients $\lambda_t^{(m)}$ is also more costly in the IAPF. The IAPF can also be explained from the two perspectives (traditional and MIS) described in Figure 2 for the APF.

Numerical example

We now demonstrate the differences in the functions used for generation of new particles among the BPF, APF, and IAPF. We use a toy example, where the model is linear and Gaussian, and where we focus on the transition of a generic PF with $M = 4$ particles from time instant $t - 1$ to t . The transition kernels are Gaussian, i.e., $p(x_t | x_{t-1}^{(m)}) = \mathcal{N}(x_t | x_{t-1}^{(m)}, \sigma_x^2)$, the likelihood function $p(y_t | x_t) = \mathcal{N}(x_t; y_t, \sigma_y^2)$ is also Gaussian, and the weights at $t - 1$ are $w_{t-1} = [0.03, 0.16, 0.16, \text{ and } 0.65]$.

Figure 3(a) displays the weighted kernels $\{w_{t-1}^{(m)} p(x_t | x_{t-1}^{(m)})\}_{m=1}^M$ and the likelihood function. The green circles represent the value of the likelihood at the center of each kernel, i.e., $\{p(y_t | \bar{x}_t^{(m)})\}_{m=1}^4$, where $\bar{x}_t^{(m)}$ is defined in (A11). Figure 3(b) shows the target distribution $p(x_t | y_{1:t}) \propto p(y_t | x_t) p(x_t | y_{1:t-1}) \approx p(y_t | x_t) \sum_{m=1}^4 w_t^{(m)} p(x_t | x_{t-1}^{(m)})$, and the resulting BPF proposal, whereas Figure 3(c) displays the same for the APF, and Figure 3(d) depicts the IAPF proposal. We recall that, in all cases, the proposal $\psi_t(\mathbf{x}_t)$ has the form of (A8), with $\lambda_t^{(m)}$ defined in Table 1 for each algorithm. In this example, the BPF proposal presents a large mismatch with the target distribution, while the IAPF mimics it very accurately. The APF achieves an intermediate performance in terms of proposal adequacy. The MIS perspective helps explain why a filter that fails at replicating well the target with the proposal $\psi_t(\mathbf{x}_t)$ will have a poor performance. See, e.g., the numerical example in [20, Sec. 4-1], where there is a sharp likelihood (because of the small noise variance) and high width of the kernels (due to large transition noise). This setup is particularly adverse for APF according to the MIS derivation in the previous

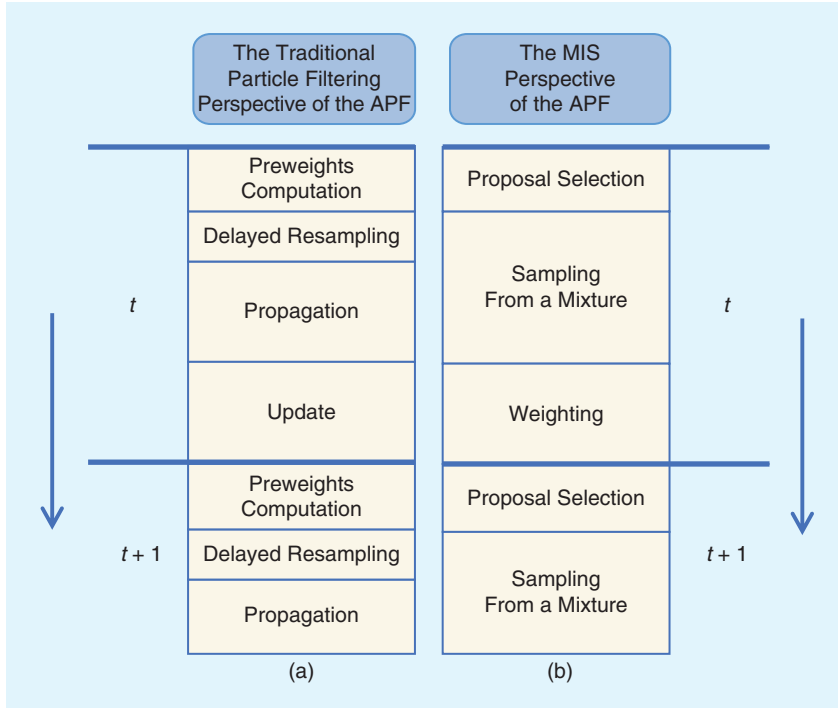


FIGURE 2. Two different perspectives for explaining the APF. (a) The traditional interpretation with the preweighting, delayed resampling, propagation, and weighting steps. (b) The MIS perspective with the proposal selection, the (mixture) sampling, and the weighting steps.

section, because the kernels will have a big overlap, and hence, the assumption for the approximation in (10) does not hold. Therefore, the proposal selection will not be close to the target, which explains the poor performance of the APF in this example.

Conclusions

In this article, we derived the APF from the perspective of MIS. With the new interpretation, we provided insights about the assumptions and approximations that are required to derive the APF. The insights are of great use for understanding when the APF does not have a satisfactory performance. The derivation also shows the relationship between the APF and BPF.

Acknowledgments

This work was supported by the Agence Nationale de la Recherche of France under award ANR-17-CE40-0031-01 and the National Science Foundation under grant CCF-1617986.

Authors

Víctor Elvira (victor.elvira@ed.ac.uk) received his B.S. degree in telecommunications engineering, M.S. degree in information technologies, and Ph.D. degree in wireless communications from the Universidad de Cantabria, Spain, in 2007, 2008, and 2011, respectively. In September 2016, he joined the Department of Communication Systems at IMT Lille Douai, France, as an associate professor. Since September 2019, he has been an associate professor (reader) with the School of Mathematics at the University of Edinburgh, Scotland, United Kingdom. He has been a visiting researcher at several institutions, including Stony Brook University, New York (as a Fulbright scholar), and Paris-Dauphine University, France. His research interests are in the field of statistical signal processing, particularly in Monte Carlo methods for Bayesian inference, stochastic optimization, and adaptive filtering, with varied applications including sensor networks, wireless communications, target tracking, and biomedicine.

Luca Martino (luca.martino@urjc.es) received his M.Sc. degree in electronic engineering from Politecnico di Milano, Italy, in 2006. In 2011, he received his Ph.D. degree in statistical signal processing from the Universidad Carlos III de Madrid, Spain, where he is currently an assistant professor in the Department of Signal Theory and Communications. He is also an assistant professor at the Universidad Rey Juan Carlos, Spain.

Mónica F. Bugallo (monica.bugallo@stonybrook.edu) received her B.S., M.S., and Ph.D. degrees in computer science and engineering from the University of A Coruña, Spain, in 1996, 1998, and 2001, respectively. She is currently a professor of electrical and computer engineering and the associate dean for Diversity and Outreach of the College of

Engineering and Applied Sciences at Stony Brook University, New York. Her research interests are in the field of statistical signal processing, with an emphasis on the theory of Monte Carlo methods and its application to different disciplines including biomedicine, ecology, sensor networks, and finance. She has dedicated her efforts to science, technology, engineering, and mathematics education and initiated several successful programs that seek to engage students at all academic stages in the excitement of engineering and research, with a focus on underrepresented groups. Currently, she is vice chair of the IEEE Signal Processing Theory and Methods Technical Committee and has served on several technical committees of IEEE conferences and workshops. She is a Senior Member of the IEEE.

Algorithm 4: The improved auxiliary particle filter.

- 1) *Initialization*: At time $t = 0$, draw M independent and identically distributed (i.i.d.) samples, $\mathbf{x}_0^{(m)}$, $m = 1, \dots, M$, from the distribution $p(\mathbf{x}_0)$.
- 2) *Recursive step*: Let $\{\mathbf{x}_{t-1}^{(m)}, w_{t-1}^{(m)}\}_{m=1}^M$ be the set of weighted particles (samples) generated at time $t-1$. At time t , proceed with the following steps:
 - Compute the mean of the probability density function $p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(m)})$ as

$$\bar{\mathbf{x}}_t^{(m)} = \mathbb{E}_{p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(m)})} [\mathbf{x}_t], \quad m = 1, \dots, M. \quad (\text{A18})$$

- Compute the normalized coefficient of each kernel in the mixture as

$$\lambda_t^{(m)} \propto \frac{p(\mathbf{y}_t | \bar{\mathbf{x}}_t^{(m)}) \sum_{j=1}^M w_{t-1}^{(j)} p(\bar{\mathbf{x}}_t^{(j)} | \mathbf{x}_{t-1}^{(j)})}{\sum_{j=1}^M p(\bar{\mathbf{x}}_t^{(j)} | \mathbf{x}_{t-1}^{(j)})}, \quad (\text{A19})$$

and select the multiple importance sampling proposal as

$$\psi^{\text{IAPF}}(\mathbf{x}_t) = \sum_{m=1}^M \lambda_t^{(m)} p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(m)}). \quad (\text{A20})$$

- Draw M i.i.d. samples from the proposal in two steps:
 - Select indexes $j^{(m)}$ and $m = 1, \dots, M$ with probability mass function given by

$$\mathbb{P}\{j^{(m)} = j\} = \lambda_t^{(j)}, \quad j \in \{1, \dots, M\}.$$

- Simulate

$$\mathbf{x}_t^{(m)} \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(j^{(m)})}), \quad m = 1, \dots, M.$$

- Compute the normalized importance sampling weights as

$$w_t^{(m)} \propto \frac{p(\mathbf{y}_t | \mathbf{x}_t^{(m)}) \sum_{j=1}^M w_{t-1}^{(j)} p(\mathbf{x}_t^{(j)} | \mathbf{x}_{t-1}^{(j)})}{\sum_{j=1}^M \lambda_t^{(j)} p(\mathbf{x}_t^{(j)} | \mathbf{x}_{t-1}^{(j)})}. \quad (\text{A21})$$

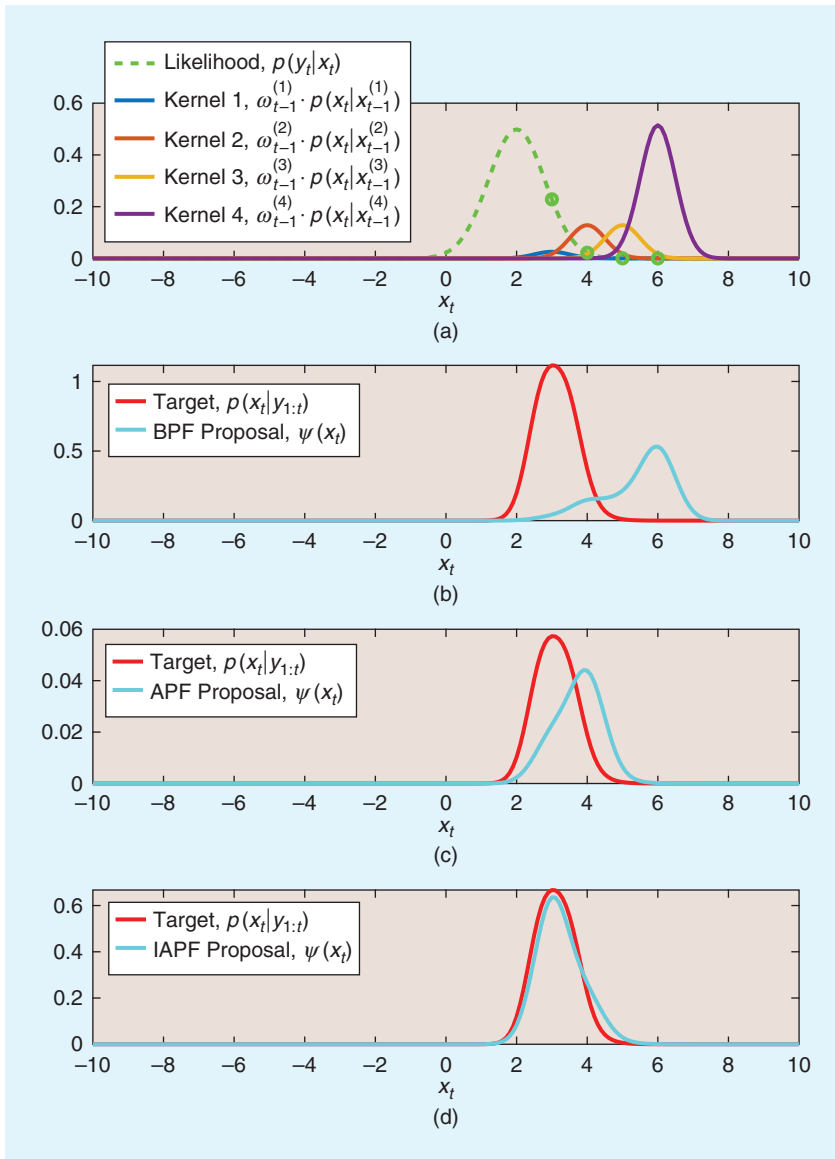


FIGURE 3. An example with $M = 4$ particles and Gaussian kernels $p(x_t | x_{t-1}^{(m)}) = \mathcal{N}(x_t | x_{t-1}^{(m)}, \sigma_x^2)$. (a) Weighted kernels $\{\tilde{w}_{t-1}^{(m)} p(x_t | x_{t-1}^{(m)})\}_{m=1}^M$ and likelihood $p(y_t | x_t) = \mathcal{N}(x_t | y_t, \sigma_y^2)$. In green circles are the values of the likelihood at the center of each kernel, i.e., $\{p(y_t | \tilde{x}_{t-1}^{(m)})\}_{m=1}^M$. (b) The target distribution $p(x_t | y_{1:t}) \propto p(y_t | x_t) p(x_t | y_{1:t-1}) \approx p(y_t | x_t) \sum_{m=1}^M \tilde{w}_{t-1}^{(m)} p(x_t | x_{t-1}^{(m)})$ and the BPF proposal ($\lambda_t^{\text{BPF}} = [0.03, 0.16, 0.16, 0.65]$). (c) The target distribution and the APF proposal ($\lambda_t^{\text{APF}} = [0.6713, 0.3221, 0.0065, 0.0001]$). (d) The target distribution and the IAPF proposal ($\lambda_t^{\text{IAPF}} = [0.7657, 0.2276, 0.0066, 0.0001]$).

Petar M. Djurić (petar.djuric@stonybrook.edu) received his B.S. and M.S. degrees in electrical engineering from the University of Belgrade, Yugoslavia, in 1981 and 1986, respectively, and his Ph.D. degree in electrical engineering from the University of Rhode Island, Kingston in 1990. He is a SUNY Distinguished Professor and serves as a chair in the Department of Electrical and Computer Engineering at Stony Brook University, New York. His

research has focused on signal and information processing. In 2012, he was a recipient of the EURASIP Technical Achievement Award. He was the first editor-in-chief of *IEEE Transactions on Signal and Information Processing Over Networks*. He is a Fellow of the IEEE and of EURASIP.

References

[1] N. Gordon, D. Salmond, and A. F. M. Smith, "Novel approach to nonlinear and non-Gaussian Bayesian state estimation," *IEE Proc.-F Radar*

Signal Process., vol. 140, no. 2, pp. 107–113, 1993.

[2] A. Doucet, N. de Freitas, and N. Gordon, "An introduction to sequential Monte Carlo methods," in *Sequential Monte Carlo Methods in Practice* (Statistics for Engineering and Information Science book series). New York: Springer, 2001, pp. 3–14. doi: 10.1007/978-1-4757-3437-9_1.

[3] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online non-linear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, 2002.

[4] P. M. Djurić, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Míguez, "Particle filtering," *IEEE Signal Process. Mag.*, vol. 20, no. 5, pp. 19–38, 2003.

[5] O. Cappé, S. J. Godsill, and E. Moulines, "An overview of existing methods and recent advances in sequential Monte Carlo," *Proc. IEEE*, vol. 95, no. 5, pp. 899–924, 2007.

[6] A. Doucet and A. M. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," in *The Oxford Handbook of Nonlinear Filtering*, vol. 12, pp. 656–704, D. Crisan and B. Rozovski, Eds. London: Oxford Univ. Press, 2009, p. 3.

[7] O. Hlinka, F. Hlawatsch, and P. M. Djurić, "Distributed particle filtering in agent networks: A survey, classification, and comparison," *IEEE Signal Process. Mag.*, vol. 30, no. 1, pp. 61–81, 2013.

[8] P. Del Moral and A. Doucet, "Particle methods: An introduction with applications," *ESAIM: Proc.*, vol. 44, pp. 1–46, Jan. 2014.

[9] X. Wang, T. Li, S. Sun, and J. Corchado, "A survey of recent advances in particle filters and remaining challenges for multitarget tracking," *Sensors*, vol. 17, no. 12, p. 2707, 2017.

[10] M. K. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *J. Amer. Statistical Assoc.*, vol. 94, no. 446, pp. 590–599, 1999.

[11] R. E. Kalman, "A new approach to linear filtering and prediction problems," *J. Basic Eng.*, vol. 82, no. 1, pp. 35–45, 1960.

[12] N. Whiteley, A. M. Johansen, and S. Godsill, "Monte Carlo filtering of piecewise deterministic processes," *J. Comput. Graphical Statist.*, vol. 20, no. 1, pp. 119–139, 2011.

[13] R. Karlsson and N. Bergman, "Auxiliary particle filters for tracking a maneuvering target," in *Proc. 39th IEEE Conf. Decision and Control*, 2000, pp. 3891–3895.

[14] P. M. Djurić, M. Vemula, and M. F. Bugallo, "Target tracking by particle filtering in binary sensor networks," *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2229–2238, 2008.

[15] M. S. Haque, S. Choi, and J. Baek, "Auxiliary particle filtering-based estimation of remaining useful life of IGBT," *IEEE Trans. Ind. Electron.*, vol. 65, no. 3, pp. 2693–2703, 2018.

[16] S. Särkkä, *Bayesian Filtering and Smoothing*. Cambridge, U.K.: Cambridge Univ. Press, 2013.

[17] L. Martino, J. Read, V. Elvira, and F. Louzada, "Cooperative parallel particle filters for on-line model selection and applications to urban mobility," *Digit. Signal Process.*, vol. 60, pp. 172–185, Jan. 2017.

[18] V. Elvira, L. Martino, D. Luengo, and M. F. Bugallo, "Generalized multiple importance sampling," *Statistical Sci.*, vol. 34, no. 1, pp. 129–155, 2019.

[19] V. Elvira, J. Míguez, and P. Djurić, "Adapting the number of particles in sequential Monte Carlo methods through an online scheme for convergence assessment," *IEEE Trans. Signal Process.*, vol. 65, no. 7, pp. 1781–1794, 2017.

[20] V. Elvira, L. Martino, M. F. Bugallo, and P. Djurić, "In search for improved auxiliary particle filters," in *Proc. 26th Eur. Signal Processing Conf. (EUSIPCO)*, 2018, pp. 1637–1641.

The Superposition Principle of Linear Time-Invariant Systems

One of the most important properties of linear time-invariant (LTI) systems is the superposition principle, which states that the overall response of an LTI system to a weighted sum of signals is simply the same weighted sum of the responses to each of the individual signals. In this “Lecture Notes” column, we will provide a deeper insight into this principle. In what follows, if the superposition principle has finitely many terms, it is called the *weak superposition principle*, whereas, if it has infinitely many terms, it is called the *strong superposition principle* (SSP).

Relevance

It is well known that the output signal of an LTI system can be expressed as the convolution of the input signal with the system’s impulse response. The derivation of this result is based on two assumptions: 1) the SSP and 2) the time invariance. However, we show that the SSP may not hold for LTI systems. Therefore, the derivation of the convolution formula is not rigorous in the signal processing literature (see [1]–[4]).

In [5], it is stated that “In deriving the convolution formula, we assume that the linearity holds for infinitely many terms. Although from the definition of linearity, it easily follows that the superposition principle is valid for finitely many terms; the extension to infinitely many terms is an added requirement.” In [6], the authors say that “With suitable conditions the superposition principle with finitely many terms holds ‘in the limit’ so that it is valid when the finite summation

is replaced by an infinite summation.” And Boggess and Narcowich [7] assert that “Since the system T is linear, we can distribute T across the infinite sum.... Even though the preceding argument is not totally rigorous, the result is true with very few restrictions on either T or the space of signals being considered.”

However, these references neither give examples to illustrate the invalidity of the SSP nor provide sufficient conditions under which the SSP is valid. In this column, we give a detailed analysis of the superposition principle. Counterexamples are provided to illustrate the invalidity of the SSP. Moreover, a necessary and sufficient condition for the SSP is derived.

Prerequisites

We use uppercase letters and uppercase calligraphy letters to represent operators and sets, respectively. Greek letters are reserved for scalars. The real and complex number fields are denoted by \mathbb{R} and \mathbb{C} , and the set of integers is denoted by \mathbb{Z} . We assume that readers are familiar with the concepts of vector space, normed space, series, and convergence; see [8]–[10] for more details. The notation $\|\cdot\|_{\mathcal{X}}$ stands for a norm on normed space \mathcal{X} . The concepts of continuous mapping and linear operator are now defined because we will study the continuity of linear systems in this article.

Definition 1 (continuous mapping)

Let \mathcal{X} and \mathcal{Y} be two normed spaces over \mathbb{C} . A mapping $T: \mathcal{X} \rightarrow \mathcal{Y}$ is said to be continuous at $x \in \mathcal{X}$ if, for every sequence $\{x_k\}_{k=1}^{\infty}$ in \mathcal{X} converging to x , the sequence $\{Tx_k\}_{k=1}^{\infty}$ converges to $Tx \in \mathcal{Y}$, i.e.,

$$\lim_{k \rightarrow \infty} (Tx_k) = T\left(\lim_{k \rightarrow \infty} x_k\right). \quad (1)$$

T is said to be continuous on \mathcal{X} if it is continuous at every point of \mathcal{X} . \square

Definition 2 (linear operator)

Let \mathcal{X} and \mathcal{Y} be two vector spaces over \mathbb{C} . A mapping $T: \mathcal{X} \rightarrow \mathcal{Y}$ is called a *linear operator* from \mathcal{X} to \mathcal{Y} if

- 1) $T(x + y) = Tx + Ty$ for all $x, y \in \mathcal{X}$
- 2) $T(\alpha x) = \alpha(Tx)$ for all $\alpha \in \mathbb{C}$ and all $x \in \mathcal{X}$.

If, in addition, $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$ and $(\mathcal{Y}, \|\cdot\|_{\mathcal{Y}})$ are two normed spaces and there exists a constant $\gamma \in \mathbb{R}$ such that for all $x \in \mathcal{X}$ we have $\|Tx\|_{\mathcal{Y}} \leq \gamma \|x\|_{\mathcal{X}}$, then T is called a *bounded linear operator*. \square

A fundamental property of linear operators is that their continuity and boundedness are equivalent to each other [10, p. 97]. Another important property is that every linear operator defined on a finite-dimensional space is continuous [10, p. 96]. To avoid the theories of Lebesgue integral and generalized functions, we mainly consider the discrete-time signals and systems. The most widely used discrete-time signal spaces are $\ell^p(\mathbb{Z})$ spaces ($p \geq 1$) defined by

$$\ell^p(\mathbb{Z}) = \{x(n) \mid \|x(n)\|_p < \infty\}, \quad (2)$$

where

$$\|x(n)\|_p = \left[\sum_{n \in \mathbb{Z}} |x(n)|^p \right]^{1/p} \quad (3)$$

is the ℓ^p norm of $x(n)$. Note that both the signal and its n th value are denoted by $x(n)$. The exact meaning of $x(n)$ should be clear from the context. When $p \rightarrow \infty$, we obtain the ℓ^∞ norm as follows:

$$\|x(n)\|_\infty = \sup_{n \in \mathbb{Z}} |x(n)|, \quad (4)$$

where sup denotes the supremum. (A set $\mathcal{A} \subset \mathbb{R}$ is said to be bounded above if there is a real number μ such that $\alpha \leq \mu$

for all $\alpha \in \mathcal{A}$, and μ is then called an *upper bound* of \mathcal{A} . If $\mathcal{A} \subset \mathbb{R}$ is bounded above, then there exists a smallest upper bound, called the *least upper bound* or *supremum* of \mathcal{A} , denoted by $\sup \mathcal{A}$ [9]. Equipped with ℓ^p norms, $(\ell^p(\mathbb{Z}), \|\cdot\|_p)$ become normed spaces for $1 \leq p \leq \infty$. Since $\ell^1(\mathbb{Z}) \subset \ell^2(\mathbb{Z}) \subset \ell^\infty(\mathbb{Z})$ [4], $(\ell^1(\mathbb{Z}), \|\cdot\|_\infty)$ and $(\ell^2(\mathbb{Z}), \|\cdot\|_\infty)$ are subspaces of $(\ell^\infty(\mathbb{Z}), \|\cdot\|_\infty)$. Similarly, $(\ell^1(\mathbb{Z}), \|\cdot\|_2)$ is a subspace of $(\ell^2(\mathbb{Z}), \|\cdot\|_2)$.

A system is an operator T that takes a signal $x(n)$ to another signal $y(n) = T[x(n)]$. It should be emphasized that $x(n)$ in the expression $T[x(n)]$ must be regarded as a whole, i.e., the value of $y(n)$ at each n may depend on $x(n)$ for all n . If

$$\delta(n) = \begin{cases} 1, & n = 0, \\ 0, & n \neq 0, \end{cases} \quad (5)$$

lies in the input space of T , the output signal of $\delta(n)$, denoted by $h(n) = T[\delta(n)]$, is called the *impulse response* of T . A system T is said to be continuous if it is a continuous mapping.

Problem statement

Every signal $x(n) \in (\ell^p(\mathbb{Z}), \|\cdot\|_p)$ with $1 \leq p < \infty$ can be expressed as

$$x(n) = \sum_{m=-\infty}^{\infty} x(m)\delta(n-m). \quad (6)$$

If we suppose that T is an LTI system with impulse response $h(n)$, then

$$T[x(n)] = T\left[\sum_{m=-\infty}^{\infty} x(m)\delta(n-m)\right]. \quad (7)$$

An important question is: can the operator T and the summation \sum be interchanged? (One of the major tasks of analysis is to study the conditions under which the order of limit and other operations, such as differentiation, integration, and summation, can be interchanged [9], [10].) In most of the signal processing literature, such as [1]–[4], it is assumed that the order of T and \sum can be interchanged. In other words, for an LTI system T , we have

$$T\left[\sum_{k=1}^{\infty} \alpha_k x_k(n)\right] = \sum_{k=1}^{\infty} \alpha_k T[x_k(n)], \quad (8)$$

which is the SSP. Therefore,

$$T[x(n)] = \sum_{m=-\infty}^{\infty} x(m)T[\delta(n-m)]. \quad (9)$$

Because T is time invariant, $T[\delta(n-m)] = h(n-m)$. Thus, we obtain the convolution formula

$$\begin{aligned} T[x(n)] &= x(n) * h(n) \\ &= \sum_{m=-\infty}^{\infty} x(m)h(n-m). \end{aligned} \quad (10)$$

If the summation \sum contains finitely many terms, T and \sum can be interchanged because of the linearity of T . However, if the summation \sum contains infinitely many terms, T and \sum cannot be interchanged arbitrarily [9], [10]. Whenever we encounter the operations of a series (a summation with infinitely many terms), we have to ask: 1) does the series converge? 2) can the order of T and \sum be interchanged?

The following example [11, p. 69] shows that (8) does not hold in general.

Example 1

Let $T: (\ell^1(\mathbb{Z}), \|\cdot\|_1) \mapsto (\ell^\infty(\mathbb{Z}), \|\cdot\|_\infty)$ be defined as

$$y(n) = T[x(n)] = \sum_{m=-\infty}^{\infty} x(m). \quad (11)$$

It can be easily verified that T is an LTI system. Define a sequence $\{x_k(n)\}_{k=1}^{\infty}$ as follows:

$$x_k(n) = \begin{cases} 1, & n = k-1, \\ -1, & n = k, \\ 0, & \text{otherwise.} \end{cases} \quad (12)$$

It is obvious that $x_k(n) \in (\ell^1(\mathbb{Z}), \|\cdot\|_1)$ for every $k \geq 1$, and we have

$$\sum_{k=1}^{\infty} T[x_k(n)] = \sum_{k=1}^{\infty} \sum_{m=-\infty}^{\infty} x_k(m) = 0. \quad (13)$$

This is because, for any $k \geq 1$, all values of $x_k(n)$ are zeros except for $x_k(k-1) = 1$ and $x_k(k) = -1$. Thus, $T[x_k(n)] = 0$ for every k . However,

$$T\left[\sum_{k=1}^{\infty} x_k(n)\right] = \sum_{m=-\infty}^{\infty} \sum_{k=1}^{\infty} x_k(m) = 1. \quad (14)$$

This is because, if $m < 0$, all values of $\{x_k(m)\}_{k=1}^{\infty}$ are zeros, so $\sum_{k=1}^{\infty} x_k(m) = 0$ for all $m < 0$. If $m = 0$, all values of $\{x_k(0)\}_{k=1}^{\infty}$ are zeros, except for $x_1(0) = 1$, so $\sum_{k=1}^{\infty} x_k(0) = 1$. If $m > 0$, all values of $\{x_k(m)\}_{k=1}^{\infty}$ are zeros, except for $x_{m-1}(m) = 1$ and $x_m(m) = -1$, so $\sum_{k=1}^{\infty} x_k(m) = 0$ for all $m > 0$. Therefore, $\sum_{k=1}^{\infty} x_k(m)$ contains one nonzero value one at $m = 0$, implying $T[\sum_{k=1}^{\infty} x_k(n)] = 1$. This example is summarized in Figure 1, which demonstrates that a linear operator T and the summation \sum with infinitely many terms cannot be interchanged, i.e., linearity does not imply the SSP. \square

Main result

Suppose that T is a linear system whose input space is \mathcal{X} . Let $\{x_k(n)\}_{k=1}^{\infty}$ be a sequence in \mathcal{X} and $\{\alpha_k\}_{k=1}^{\infty}$ a sequence in \mathbb{C} such that the series $\sum_{k=1}^{\infty} \alpha_k x_k(n)$ converges to $x(n) \in \mathcal{X}$. Then, the sequence of partial sums $s_k(n) = \sum_{i=1}^k \alpha_i x_i(n)$ converges to $x(n)$, i.e.,

| | |
|------------------------------------------------------------------------------|-------------------------------------------------------------------------------|
| $x_1(n) = \cdots 0 \quad 1 \quad -1 \quad 0 \quad \cdots$ | $T[x_1(n)] = \cdots 0 \quad \cdots$ |
| $x_2(n) = \cdots 0 \quad 0 \quad 1 \quad -1 \quad \cdots$ | $T[x_2(n)] = \cdots 0 \quad \cdots$ |
| $x_3(n) = \cdots 0 \quad 0 \quad 0 \quad 1 \quad \cdots$ | $T[x_3(n)] = \cdots 0 \quad \cdots$ |
| \vdots | \vdots |
| $\sum_{k=1}^{\infty} x_k(n) = \cdots 0 \quad 1 \quad 0 \quad 0 \quad \cdots$ | $T\left[\sum_{k=1}^{\infty} x_k(n)\right] \neq \sum_{k=1}^{\infty} T[x_k(n)]$ |

FIGURE 1. If we sum the rows first and then the column, we get zero; while if we sum the columns first and then the row, we get one.

$$\lim_{k \rightarrow \infty} s_k(n) = x(n) = \sum_{i=1}^{\infty} \alpha_i x_i(n). \quad (15)$$

If T is continuous, it follows from definition 1 that

$$T[x(n)] = \lim_{k \rightarrow \infty} T[s_k(n)]. \quad (16)$$

Because T is linear,

$$T[s_k(n)] = \sum_{i=1}^k \alpha_i T[x_i(n)]. \quad (17)$$

Therefore, (16) is equivalent to (8), and we can conclude that, if T is a continuous linear system, and the series $\sum_{k=1}^{\infty} \alpha_k x_k(n)$ converges, then the principle of superposition is valid for both finitely many and infinitely many terms.

Conversely, if a system T satisfies (8) for every convergent series $\sum_{k=1}^{\infty} \alpha_k x_k(n)$, it is linear and continuous. The proof is given next. Let $\alpha_k = 0$ for $k > 2$, then, (8) becomes

$$T[\alpha_1 x_1(n) + \alpha_2 x_2(n)] = \alpha_1 T[x_1(n)] + \alpha_2 T[x_2(n)], \quad (18)$$

which shows that T is linear. In addition, for every sequence $\{x_k(n)\}_{k=1}^{\infty}$ in \mathcal{X} that converges to $x(n) \in \mathcal{X}$, we can construct another sequence $\{u_k(n)\}_{k=1}^{\infty}$ in \mathcal{X} as follows:

$$u_k(n) = \begin{cases} x_1(n), & k = 1, \\ x_k(n) - x_{k-1}(n), & k > 1. \end{cases} \quad (19)$$

Then, $x_k(n) = \sum_{i=1}^k u_i(n)$ are the partial sums of $\{u_k(n)\}_{k=1}^{\infty}$, and the series $\sum_{k=1}^{\infty} u_k(n)$ converges to $x(n)$. By the linearity of T , we have

$$T[x_k(n)] = \sum_{i=1}^k T[u_i(n)]. \quad (20)$$

Thus,

$$\lim_{k \rightarrow \infty} T[x_k(n)] = \sum_{i=1}^{\infty} T[u_i(n)]. \quad (21)$$

On the other hand, it follows from (8) that

$$\begin{aligned} \sum_{i=1}^{\infty} T[u_i(n)] &= T\left[\sum_{i=1}^{\infty} u_i(n)\right] \\ &= T\left[\lim_{k \rightarrow \infty} x_k(n)\right]. \end{aligned} \quad (22)$$

Therefore, T is continuous. Because continuity and boundedness are equivalent for a linear operator, the preceding analysis can be summarized by Theorem 1.

Theorem 1 (SSP)

Let T be a system defined on a normed space \mathcal{X} over \mathbb{C} . Suppose that $\{x_k(n)\}_{k=1}^{\infty}$ is any sequence in \mathcal{X} and $\{\alpha_k\}_{k=1}^{\infty}$ is any sequence in \mathbb{C} such that the series $\sum_{k=1}^{\infty} \alpha_k x_k(n)$ converges in \mathcal{X} . Then, the SSP (8) holds if and only if T is linear and continuous, or, equivalently, if T is linear and bounded. \square

It should be pointed out that theorem 1 is a very general result. It holds for continuous-time signals and systems, too. If the linearity and time invariance imply the continuity, then the SSP holds for any LTI system. Nevertheless, this is not true; see Example 2.

Example 2

Let T be the accumulator (an LTI system [3]) given by

$$y(n) = T[x(n)] = \sum_{m=-\infty}^n x(m). \quad (23)$$

Since $|y(n)| \leq \|x(m)\|_1$ for each n , T is an LTI system from $\ell^1(\mathbb{Z})$ to $\ell^\infty(\mathbb{Z})$. We endow $\ell^1(\mathbb{Z})$ and $\ell^\infty(\mathbb{Z})$ with the norms of $\|\cdot\|_2$ and $\|\cdot\|_\infty$, respectively. It should be noted that the normed spaces $(\ell^1(\mathbb{Z}), \|\cdot\|_1)$ and $(\ell^1(\mathbb{Z}), \|\cdot\|_2)$ are different because their norms are defined differently. Since $\ell^1(\mathbb{Z}) \subset \ell^2(\mathbb{Z})$, $(\ell^1(\mathbb{Z}), \|\cdot\|_2)$ is a subspace of $(\ell^2(\mathbb{Z}), \|\cdot\|_2)$. Let $\{x_k(n)\}_{k=1}^{\infty}$ be a sequence defined as

$$x_k(n) = \begin{cases} \frac{1}{n}, & 1 \leq |n| \leq k, \\ 0, & \text{otherwise.} \end{cases} \quad (24)$$

It is obvious that $x_k(n) \in (\ell^1(\mathbb{Z}), \|\cdot\|_2)$ for every $k \geq 1$. Because the series $\sum_{n=1}^{\infty} 1/n^2$ converges [9, p. 62], $\|x_k(n)\|_2$ increases monotonically to a fixed real number as $k \rightarrow \infty$. Whereas, because the series $\sum_{n=1}^{\infty} 1/n$ diverges [9, p. 62],

$$\|T[x_k(n)]\|_\infty = |y_k(0)| = \sum_{n=1}^k \frac{1}{n} \quad (25)$$

tends to ∞ as $k \rightarrow \infty$. Hence, there exists no real number γ such that $\|T[x_k(n)]\|_\infty \leq \gamma \|x_k(n)\|_2$ for all k , which is to say, T is unbounded (= discontinuous). \square

This example shows that the linearity and time invariance do not imply the SSP. To better understand theorem 1, here are two additional examples.

Example 3

Let T be a linear filter whose output is computed by (10). If its impulse response $h(n)$ is absolutely summable, i.e., $h(n) \in \ell^1(\mathbb{Z})$, then it follows from the Young's inequality [11, p. 240] that for any $x(n) \in \ell^p(\mathbb{Z})$ with $1 \leq p \leq \infty$, we have

$$\begin{aligned} \|T[x(n)]\|_p &= \|h(n) * x(n)\|_p \\ &\leq \|h(n)\|_1 \|x(n)\|_p. \end{aligned} \quad (26)$$

Thus, T is a bounded linear operator from $(\ell^p(\mathbb{Z}), \|\cdot\|_p)$ to $(\ell^p(\mathbb{Z}), \|\cdot\|_p)$. And, by theorem 1, we know that the SSP holds for a linear filter if its impulse response is absolutely summable. \square

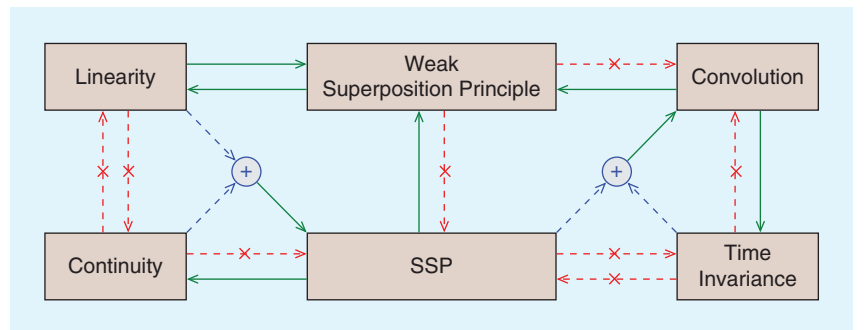


FIGURE 2. The relations among linearity, continuity, time invariance, convolution, and superposition principles.

Example 4

Differentiation is a basic LTI system and is widely used in signal processing. However, it is discontinuous in the norm $\|x(t)\| = \sup_{t \in \mathbb{R}} |x(t)|$ [10, p. 93]. Thus, by theorem 1, it does not possess the SSP. For example, let

$$x_k(t) = \begin{cases} \sin(t), & k = 1, \\ \frac{\sin(kt)}{k} - \frac{\sin(kt-t)}{k-1}, & k > 1. \end{cases} \quad (27)$$

Then, $\sum_{k=1}^{\infty} x_k(t) = \lim_{q \rightarrow \infty} [\sin(qt)/q]$, which converges to zero, and we have $(d/dt) \sum_{k=1}^{\infty} x_k(t) = 0$. However, $\sum_{k=1}^{\infty} (d/dt) x_k(t) = \lim_{q \rightarrow \infty} \cos(qt) \neq 0$. Therefore,

$$\frac{d}{dt} \sum_{k=1}^{\infty} x_k(t) \neq \sum_{k=1}^{\infty} \frac{d}{dt} x_k(t). \quad (28)$$

Hence, we cannot arbitrarily interchange the differential operator and the summation. \square

The relations among linearity, continuity, time invariance, convolution, and superposition principles are summarized in Figure 2. Because the linearity and time invariance do not ensure the SSP, (8) does not hold in general, and, hence, the convolution formula (10) may not hold either. However, in most of the literature, it is stated that “the output signal of an LTI system can be computed by the convolution of the input signal with the system’s impulse response.”

Now we know that this conclusion is not true in general.

What we have learned

For an LTI system T , the SSP may not hold because the operator T and the summation Σ cannot be arbitrarily interchanged. Therefore, the derivation of convolution formulas is not rigorous in the literature of signal processing. A sufficient condition that ensures the interchangeability of T and Σ is that T is continuous. When the input space of T is finite dimensional, the SSP automatically holds. Considering that all sequences in the real world have finite dimensions, this lack of mathematical rigor will not have a substantive impact on practical applications.

Acknowledgments

We thank the anonymous reviewers for their helpful comments and suggestions. This work was supported in part by the National Natural Science Foundation of China under grant 61801368 and in part by the China Postdoctoral Science Foundation under grant 2018M640990.

Authors

Ming Zhang (itjerry@163.com) received his Ph.D. degree in electronic science and technology from Xi’an Jiaotong University, China, in 2017. He is currently a postdoctoral researcher with the School of Information and Communications Engineering, Xi’an

Jiaotong University. His research interests include adaptive filtering, spectrum estimation, and beamforming.

Anxue Zhang (anxuezhang@mail.xjtu.edu.cn) received his Ph.D. degree in electronic science and technology from Xi’an Jiaotong University, China, in 2003. He is currently a professor with the School of Information and Communications Engineering, Xi’an Jiaotong University. His research interests include electromagnetic wave theory and antenna design.

References

- [1] S. Haykin and B. V. Veen, *Signals and Systems*, 2nd ed. Hoboken, NJ: Wiley, 2003.
- [2] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing*, 4th ed. Englewood Cliffs, NJ: Prentice Hall, 2006.
- [3] A. V. Oppenheim and R. W. Schaffer, *Discrete-Time Signal Processing*, 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 2010.
- [4] M. Vetterli, J. Kovačević, and V. K. Goyal, *Foundations of Signal Processing*. Cambridge, U.K.: Cambridge Univ. Press, 2014.
- [5] A. Papoulis, *The Fourier Integral and Its Applications*. New York: McGraw-Hill, 1962.
- [6] D. B. Percival and A. T. Walden, *Spectral Analysis for Physical Applications*. Cambridge, U.K.: Cambridge Univ. Press, 1993.
- [7] A. Boggess and F. J. Narcowich, *A First Course in Wavelets With Fourier Analysis*, 2nd ed. Hoboken, NJ: Wiley, 2009.
- [8] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*. Philadelphia: SIAM, 2000.
- [9] W. Rudin, *Principles of Mathematical Analysis*, 3rd ed. New York: McGraw-Hill, 1976.
- [10] E. Kreyszig, *Introductory Functional Analysis with Applications*. New York: Wiley, 1978.
- [11] G. B. Folland, *Real Analysis: Modern Techniques and Their Applications*, 2nd ed. New York: Wiley, 1999.

SP

Are You Moving?



Don't miss an issue of this magazine—
update your contact information now!

Update your information by:

E-MAIL: address-change@ieee.org

PHONE: +1 800 678 4333 in the United States
or +1 732 981 0060 outside
the United States

If you require additional assistance
regarding your IEEE mailings,
visit the IEEE Support Center
at supportcenter.ieee.org.



© ISTOCKPHOTO.COM/BRIANAJACKSON

Ibrahim Al-Nahhal, Octavia A. Dobre, Ertugrul Basar,
Cecilia Moloney, and Salama Ikki

A Fast, Accurate, and Separable Method for Fitting a Gaussian Function

The Gaussian function (GF) is widely used to explain the behavior or statistical distribution of many natural phenomena as well as industrial processes in different disciplines of engineering and applied science. For example, the GF can be used to model an approximation of the Airy disk in image processing, a laser heat source in laser transmission welding [1], practical microscopic applications [2], and fluorescence dispersion in flow cytometric deoxyribonucleic acid histograms [3]. In applied sciences, the noise that corrupts the signal can be modeled by the Gaussian distribution according to the central limit theorem. Thus, by fitting the GF, researchers can develop a sound interpretation of the corresponding process or phenomenon behavior.

This article introduces a novel fast, accurate, and separable (FAS) algorithm for estimating the GF parameters to fit observed data points. A simple mathematical trick can be used to calculate the area under the GF in two ways. Then, by equating these two areas, the GF parameters can be easily obtained from the observed data.

GF-fitting approaches

A GF has a symmetrical bell shape around its center, with a width that smoothly decreases as it moves away

from its center on the x -axis. The mathematical form of the GF is

$$y(x) = Ae^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad (1)$$

with three shape-controlling parameters, A , μ , and σ , where A is the maximum height (amplitude) that can be achieved on the y -axis, μ is the curve center (mean) on the x -axis, and σ is the standard deviation (SD), which controls the width of the curve along the x -axis. This article presents a new method for the accurate estimation of these three parameters. The difficulty of this lies in estimating the three shape-controlling parameters (A , μ , and σ) from observations, which are generally noisy, by solving an overdetermined nonlinear system of equations.

The standard solutions for fitting the GF parameters from noisy observed data are obtained by one of the following two approaches:

- 1) *Solving the problem as a nonlinear system of equations using one of the least-squares optimization algorithms:* This solution employs an iterative procedure, such as the Newton–Raphson algorithm [4]. The drawbacks of this approach are the iterative procedure, which may not converge to the true solution, and its high cost from the computational complexity perspective.
- 2) *Solving the problem as a linear system of equations based on the fact*

that the GF is an exponential of a quadratic function: By taking the natural logarithm of the observed data, the problem can be solved in polynomial time as a 3×3 linear system of equations. Two traditional algorithms have been proposed in this context: Caruana’s algorithm [5] and Guo’s algorithm [6]. Furthermore, instead of taking the natural logarithm, the partial derivative is used in Roonizi’s algorithm [7].

In this article, we consider only the second approach, which is more suitable for most scientific applications, due to its simplicity and because it avoids the drawbacks of the first approach. Let us start with a brief introduction of the existing three algorithms for the second approach.

Caruana’s algorithm

Caruana’s algorithm exploits the fact that the GF is an exponential of a quadratic function and transforms it into a linear form by taking the natural logarithm of (1) to obtain

$$\begin{aligned} \ln(y) &= \ln(A) + \frac{-(x-\mu)^2}{2\sigma^2} \\ &= \ln(A) - \frac{\mu^2}{2\sigma^2} + \frac{2\mu x}{2\sigma^2} - \frac{x^2}{2\sigma^2} \\ &= a + bx + cx^2, \end{aligned} \quad (2)$$

where $a = \ln(A) - \mu^2/(2\sigma^2)$, $b = \mu/\sigma^2$, and $c = -1/(2\sigma^2)$. Accordingly, the

unknowns become a , b , and c in the linear equation (2) instead of A , μ , and σ in the nonlinear equation (1). Next, if the observations y are noisy, then they can be modeled as $\hat{y} = y + w$. Each contains the ideal data point, y , that is corrupted by the noise, w , with SD of σ_w . Note that in (2), we consider only the observations that have values above zero.

Once we have an overdetermined linear system, the unknowns can be estimated using the least-squares method. Caruana's algorithm estimates the three unknowns (a , b , and c) in (2) using the least-squares method by forming the error function, ε , for (2) as

$$\begin{aligned}\varepsilon &= \ln(\hat{y}) - \ln(y) \\ &= \ln(\hat{y}) - (a + bx + cx^2).\end{aligned}\quad (3)$$

Then, by differentiating the sum of ε^2 with respect to a , b , and c and equating the results to zero, we obtain three equations, which represent the following linear system:

$$\begin{bmatrix} N & \sum x_n & \sum x_n^2 \\ \sum x_n & \sum x_n^2 & \sum x_n^3 \\ \sum x_n^2 & \sum x_n^3 & \sum x_n^4 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum \ln(\hat{y}_n) \\ \sum x_n \ln(\hat{y}_n) \\ \sum x_n^2 \ln(\hat{y}_n) \end{bmatrix}, \quad (4)$$

where N is the number of observed data points and \sum denotes $\sum_{n=1}^N$. In this case, the parameters a , b , and c can be determined simply by solving (4) as a determined linear system of equations. Subsequently, the original parameters of the GF are determined as

$$A = e^{a - \frac{b^2}{4c}}, \quad \mu = \frac{-b}{2c}, \quad \sigma = \sqrt{\frac{-1}{2c}}. \quad (5)$$

The weighted least-squares method is the second candidate method to estimate the unknowns, and it is expected to have a better estimation accuracy than the least-squares method.

Guo's algorithm

Guo's algorithm, a modified version of the Caruana algorithm, finds the unknowns a , b , and c in (2) using the

weighted least-squares method. It uses the noisy observed data, \hat{y} , to weight the error function in (3). Therefore, the error equation in (3) becomes $\delta = \hat{y}\varepsilon = \hat{y}[\ln(\hat{y}) - (a + bx + cx^2)]$, and the linear system of equations in (4) becomes

$$\begin{bmatrix} \sum \hat{y}_n^2 & \sum x_n \hat{y}_n^2 & \sum x_n^2 \hat{y}_n^2 \\ \sum x_n \hat{y}_n^2 & \sum x_n^2 \hat{y}_n^2 & \sum x_n^3 \hat{y}_n^2 \\ \sum x_n^2 \hat{y}_n^2 & \sum x_n^3 \hat{y}_n^2 & \sum x_n^4 \hat{y}_n^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum \hat{y}_n^2 \ln(\hat{y}_n) \\ \sum x_n \hat{y}_n^2 \ln(\hat{y}_n) \\ \sum x_n^2 \hat{y}_n^2 \ln(\hat{y}_n) \end{bmatrix}. \quad (6)$$

Moreover, the values of A , μ , and σ can be computed from (5).

One of the problems that affects the estimation accuracy is the long-tail GF. This occurs when the number of small values in the observed data is large compared to the observed data length, N , which means that a large amount of noise exists in those observations. Thus, an iterative procedure is required to improve the estimation accuracy.

Guo's algorithm with iterative procedure

The estimation accuracy of the Guo's algorithm deteriorates for a long-tail GF. To increase the accuracy of fitting the long-tail Gaussian parameters, an iterative procedure for (6) is given as

$$\begin{bmatrix} \sum \hat{y}_{n,(k-1)}^2 & \sum x_n \hat{y}_{n,(k-1)}^2 & \sum x_n^2 \hat{y}_{n,(k-1)}^2 \\ \sum x_n \hat{y}_{n,(k-1)}^2 & \sum x_n^2 \hat{y}_{n,(k-1)}^2 & \sum x_n^3 \hat{y}_{n,(k-1)}^2 \\ \sum x_n^2 \hat{y}_{n,(k-1)}^2 & \sum x_n^3 \hat{y}_{n,(k-1)}^2 & \sum x_n^4 \hat{y}_{n,(k-1)}^2 \end{bmatrix} \times \begin{bmatrix} a^{(k)} \\ b^{(k)} \\ c^{(k)} \end{bmatrix} = \begin{bmatrix} \sum \hat{y}_{n,(k-1)}^2 \ln(\hat{y}_n) \\ \sum x_n \hat{y}_{n,(k-1)}^2 \ln(\hat{y}_n) \\ \sum x_n^2 \hat{y}_{n,(k-1)}^2 \ln(\hat{y}_n) \end{bmatrix}, \quad (7)$$

where $\hat{y}_{n,(k)} = \hat{y}_n$ for $k = 0$ and $\hat{y}_{n,(k)} = e^{a^{(k)} + b^{(k)}x_n + c^{(k)}x_n^2}$ for $k > 0$, with the parenthesized subscripts denoting the indices of iteration.

Roonizi's algorithm

Roonizi's algorithm is designed to fit the GF riding on a polynomial background. It can be used to fit a GF by taking the partial derivative of (1), and then taking the integral of the result to obtain

$$y(x) = \beta_1 \phi_1(x) + \beta_2 \phi_2(x), \quad (8)$$

where $\beta_1 = -1/\sigma^2$, $\beta_2 = \mu/\sigma^2$, and

$$\phi_1(x) = \int_{-\infty}^x uy(u)du, \quad \phi_2(x) = \int_{-\infty}^x y(u)du. \quad (9)$$

In a manner similar to the steps in the Caruana and Guo algorithms, the error of (8) becomes $\zeta = \hat{y} - (\beta_1 \phi_1(x) + \beta_2 \phi_2(x))$. A linear system of equations results as follows:

$$\begin{bmatrix} \sum |\phi_1(x_n)|^2 & \sum \phi_1(x_n) \phi_2(x_n) \\ \sum \phi_1(x_n) \phi_2(x_n) & \sum |\phi_2(x_n)|^2 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \sum \phi_1(x_n) \hat{y}_n \\ \sum \phi_2(x_n) \hat{y}_n \end{bmatrix}. \quad (10)$$

By solving (10) in terms of β_1 and β_2 , the estimated $\hat{\mu}$ and $\hat{\sigma}$ of the GF can be calculated as

$$\hat{\sigma} = \sqrt{\frac{-1}{\beta_1}}, \quad \hat{\mu} = \frac{-\beta_2}{\beta_1}. \quad (11)$$

Finally, using $\hat{\mu}$ and $\hat{\sigma}$ from (11), the estimated \hat{A} of the GF can be calculated as

$$\hat{A} = \frac{\sum \left(\hat{y}_n \exp\left(\frac{-(x_n - \hat{\mu})^2}{2\hat{\sigma}^2}\right) \right)}{\sum \exp\left(\frac{-(x_n - \hat{\mu})^2}{2\hat{\sigma}^2}\right)}. \quad (12)$$

Note that the Roonizi's algorithm has no iterative procedure to increase the accuracy of fitting long-tail GF parameters.

Motivation

Guo's and Roonizi's algorithms have better estimation accuracy than Caruana's algorithm, while their computational complexity burden is comparable. Moreover, the three algorithms dependently estimate the GF parameters (A , μ , and σ). This means that, in some applications that require the estimation of only one parameter, the fitting algorithm may require unnecessary parameters to be estimated as well. Therefore, there is a need for a new method that provides better estimation accuracy with an efficient computational complexity as well as

the capability for a separable parameter estimation.

Proposed algorithm

In this article, we propose a novel FAS algorithm for a GF that accurately fits the observed data. The basic idea of the proposed FAS algorithm is to find a direct formula for the SD (i.e., σ) parameter from the noisy observed data. Then the amplitude A and mean μ can be determined using the weighted least-squares method for only two unknowns.

Derivation of the SD formula

To derive an approximation formula for the SD, a simple mathematical trick is applied. For N observations that represent the GF, as shown in Figure 1, the area under the GF can be divided into thin vertical rectangles with a width of Δx_n , where Δx_n is the n th step size of two successive observation points on the x -axis. Therefore, the total area under the GF, Λ , is numerically calculated as the summation of the areas of the vertical rectangles:

$$\Lambda \approx \sum_{n=1}^N \Delta x_n \hat{y}_n. \quad (13)$$

Note that (13) reflects at least 99.7% of the GF area in case of an available observation width greater than $\mu \pm 3\sigma$. Now, let us calculate the area under the GF using a different method. From the GF and Q -function properties, the total area under the GF is given as

$$\Lambda = \int_{-\infty}^{\infty} A e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx = A\sigma\sqrt{2\pi}. \quad (14)$$

Equating (13) and (14), and replacing the amplitude A by the maximum value of the observed data, \hat{y}_{\max} , the estimated σ is obtained as

$$\hat{\sigma} = \frac{\sum_{n=1}^N \Delta x_n \hat{y}_n}{\sqrt{2\pi} \hat{y}_{\max}}. \quad (15)$$

Thus, in certain applications that require the estimate of the SD of the GF, the FAS

algorithm directly outputs this estimate, without estimating the other two parameters. This is referred to as the *separable property* of the FAS algorithm.

Error analysis

To study the error of (15), first let us discuss the systematic error resulting from equating (13) and (14). This error becomes notable when a small portion of the GF is sampled, and the GF curve is approximated by rectangles (as in Figure 1). Based on extensive testing of the algorithm with varying parameters, as discussed further later in the article, the systematic error can be considered negligible when $W > 6$ and the observation samples are dense enough [e.g., $(N/W) > 10$], where W is the ratio of the SD to the observation width on the x -axis (i.e., the observation width equals $W\sigma$, or equivalently, it varies from $\mu - (W/2)\sigma$ to $\mu + (W/2)\sigma$).

To calculate the relative error in the numerator in (15), let the numerator equal $\sqrt{2\pi} A\sigma + \Delta x \sum_{n=1}^N w_n$, where $\sqrt{2\pi} A\sigma$ represents the actual area of the GF and $\Delta x \sum_{n=1}^N w_n$ is normally distributed with its SD being $\sqrt{N} \sigma_w \Delta x = \sqrt{N} \sigma_w (W\sigma/N)$. For simplicity of analysis, Δx is considered fixed for all observations. The relative error of the numerator, α_N , can be written as

$$\alpha_N \approx k_1 \frac{\sigma_w W}{\sqrt{2\pi} A \sqrt{N}} = k_1 \frac{W}{\text{SNR} \sqrt{2\pi N}}, \quad (16)$$

where k_1 is a constant value, which can be considered 2 for the 95.5% confidence interval, and $\text{SNR} = A/\sigma_w$ is the signal-to-noise ratio (SNR).

For the denominator, let us assume that it equals $\sqrt{2\pi} (A \pm \Delta A)$, where ΔA is the maximum of the normally distributed noise samples with SD of σ_w . The relative error of the denominator in (15), α_D , can be written as

$$\alpha_D \approx \frac{k_2 \sigma_w}{A} = \frac{k_2}{\text{SNR}}, \quad (17)$$

where k_2 is a constant whose value can be assumed to be 3. (Based on com-

prehensive simulations, $k_2 = 3$ is the worst-case scenario for the error. Also, the probability of such a scenario is very low.) Hence, the total relative error in (15), α , can be approximated using a Taylor series as

$$\alpha \approx \alpha_N + \alpha_D = \frac{1}{\text{SNR}} \left(k_1 \frac{W}{\sqrt{2\pi N}} + k_2 \right). \quad (18)$$

If the samples are dense enough (i.e., large enough N/W), a reduced relative error can be attained for a high SNR.

Estimates of the remaining two parameters

To estimate the remaining two parameters A and μ using $\hat{\sigma}$ estimated from (15), we can differentiate the sum of δ^2 with respect to a and b and then equate the results to zero (i.e., using the same steps as in Guo's algorithm). The resulting linear system of equations becomes

$$\begin{bmatrix} \sum \hat{y}_n^2 & \sum x_n \hat{y}_n^2 \\ \sum x_n \hat{y}_n^2 & \sum x_n^2 \hat{y}_n^2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum \hat{y}_n^2 \ln(\hat{y}_n) - c \sum x_n^2 \hat{y}_n^2 \\ \sum x_n \hat{y}_n^2 \ln(\hat{y}_n) - c \sum x_n^3 \hat{y}_n^2 \end{bmatrix}, \quad (19)$$

where $c = -1/(2\hat{\sigma}^2)$ and $\hat{\sigma}$ is the estimated SD, which is calculated from (15). Therefore, the values of a and b are obtained by solving the 2×2 linear system in (19); then, the original parameters A and μ can be calculated from (5).

Figure 2 shows the superiority of the proposed FAS algorithm over the traditional algorithms in the presence of a noise with SD $\sigma_w = 0.1$ for different values of N ; the proposed algorithm

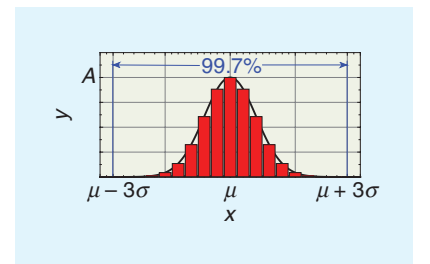


FIGURE 1. A graph illustrating a Gaussian function.

provides the best fit to the observed data points compared to the other fitting algorithms for all values of N . Figure 2 shows that \hat{y}_{\max} is obviously different from the actual amplitude A . However, $\hat{\sigma}$ from (15) provides reasonable results using \hat{y}_{\max} even if a small number of observation points are available, as in Figure 2(c).

Since the FAS algorithm provides poorer accuracy in fitting long-tail GF parameters, an iterative procedure is required to improve the fitting accuracy.

FAS algorithm with iterative procedure

For the long-tail GF, we propose an iterative algorithm that improves the

fitting accuracy of the FAS algorithm. The recursive version of (19) is given as

$$\begin{bmatrix} \sum \hat{y}_{n,(k-1)}^2 & \sum x_n \hat{y}_{n,(k-1)}^2 \\ \sum x_n \hat{y}_{n,(k-1)}^2 & \sum x_n^2 \hat{y}_{n,(k-1)}^2 \end{bmatrix} \begin{bmatrix} a(k) \\ b(k) \end{bmatrix} = \begin{bmatrix} \sum \hat{y}_{n,(k-1)} \ln(\hat{y}_n) - c \sum x_n^2 \hat{y}_{n,(k-1)}^2 \\ \sum x_n \hat{y}_{n,(k-1)} \ln(\hat{y}_n) - c \sum x_n^3 \hat{y}_{n,(k-1)}^2 \end{bmatrix}, \quad (20)$$

where $\hat{y}_{n,(k)} = \hat{y}_n$ for $k = 0$, $\hat{y}_{n,(k)} = e^{a(k) + b(k)x_n + cx_n^2}$ for $k > 0$, and $\hat{\sigma}$ is estimated from (15) only once. This means that (15) can provide accurate results in fitting the long-tail GF without iteration, while the other two parameters still need to be estimated through iterations.

However, after a few iterations, $\hat{\sigma}$ can be further improved by including an updated SD from (15) in the iterations, using A obtained by (20).

Figure 3 shows results of the iterative Guo's and proposed FAS algorithms for fitting a long-tail GF with $N = 200$, $A = 1$, $\sigma = 2$, and $\sigma_w = 0.1$ for $\mu = 18$ and 19, respectively. As we can see from the figure, the number of iterations required for the FAS algorithm to fit the long-tail GF is lower than that for Guo's algorithm. For example, in Figure 3(a), the FAS algorithm needs only three iterations to fit the observation; however, Guo's algorithm provides poor fitting for the same number of iterations. Note that, from Figure 3(b), as the tail of the

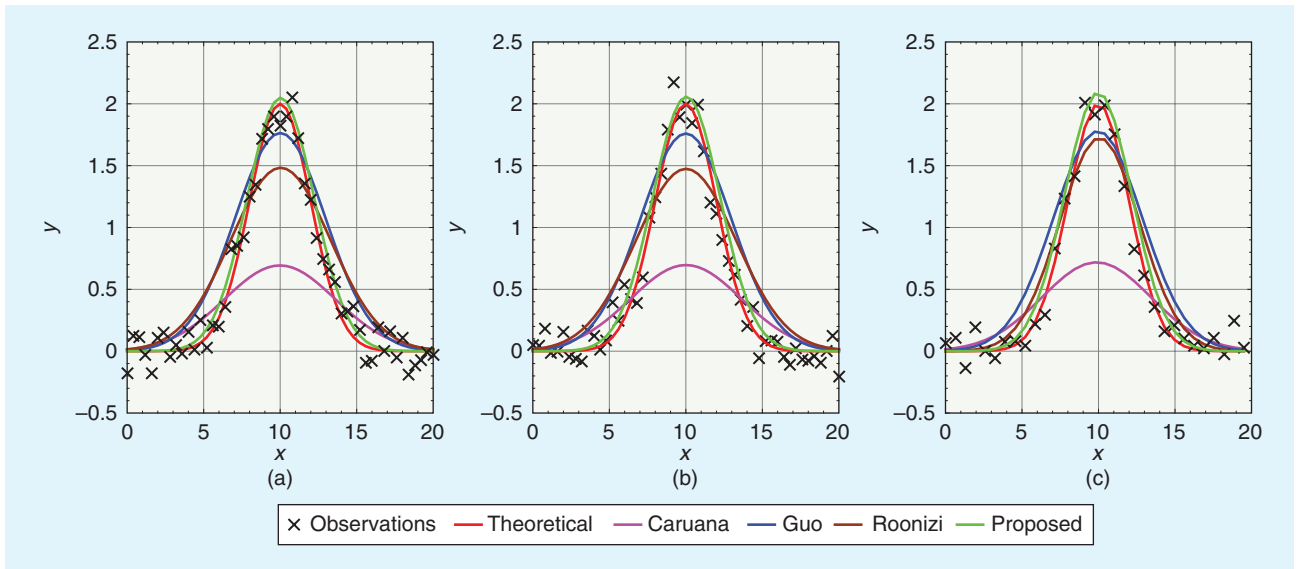


FIGURE 2. Graphs showing the results of different algorithms for fitting the GF with $A = 2$, $\sigma = 2$, and $\mu = 10$ in the presence of observation noise with $\sigma_w = 0.1$ (i.e., $\text{SNR} = 10$). (a) $N = 50$. (b) $N = 40$. (c) $N = 30$.

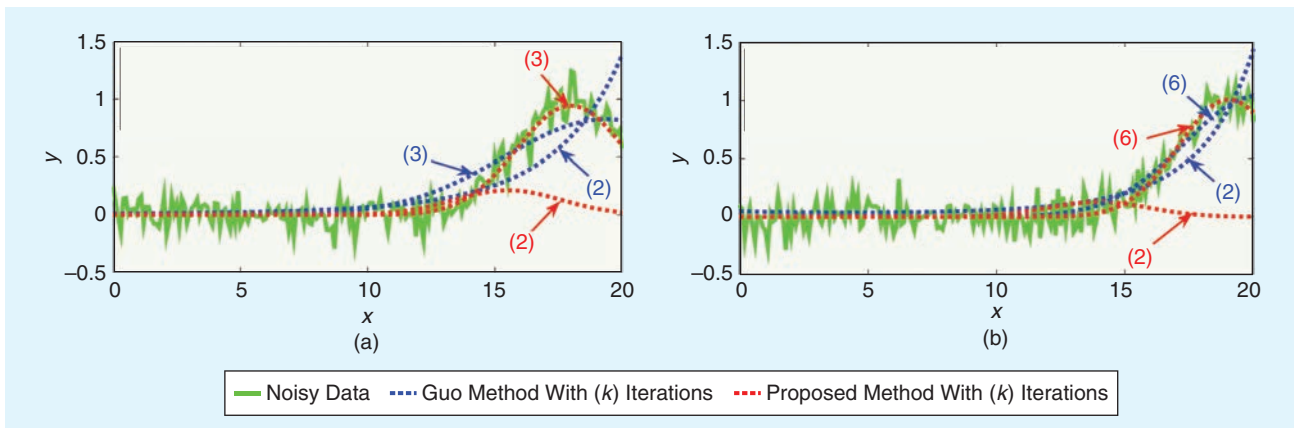


FIGURE 3. Graphs showing the results of the proposed FAS iterative algorithm in comparison with Guo's algorithm for fitting the GF of $N = 200$, $A = 1$, $\sigma = 2$, and $\sigma_w = 0.1$ (i.e., $\text{SNR} = 10$). (a) $\mu = 18$. (b) $\mu = 19$.

GF lengthens, more iterations are needed (i.e., six iterations are needed instead of three to provide a good fitting to the longer-tail GF). Even in the presence of considerable noise and with only a small portion of the GF, the iterative procedure of the proposed algorithm can nicely fit the GF after only a few iterations.

Accuracy comparison

In this section, Monte Carlo simulation results for at least 10^4 simulated trials are considered for comparing the average absolute relative error (ARE) of the fitting accuracy for the SD estimated using (15) and by traditional algorithms. The ARE percentile of the SD is given as $\text{ARE\%}(\sigma) = (|\hat{\sigma} - \sigma|/\sigma) \times 100\%$, where $|\cdot|$ denotes the absolute value and σ is the true SD. The GF parameters used for this simulation are $A = 1$, $\mu = 10$, and $\sigma = 2$. As demonstrated by the total relative error estimated in (18), three parameters can be used for assess-

ing the accuracy of estimation (i.e., SNR, W , and N).

For the evaluation of the estimation accuracy, we calculate the average $\text{ARE\%}(\sigma)$, where one of the three parameters varies while the other two parameters are fixed. Figure 4 shows such results, where the SD is estimated using the proposed FAS algorithm in comparison with the three previously presented traditional algorithms. In Figure 4(a) and (b), $W = 12$ and the SNR varies from 1 to 100 for $N = 30$ and 200, respectively. Figure 4(c) and (d) depicts the effect of W , which varies from 2 to 24, for $N = 30$ and 200, respectively, in the case of SNR = 25 (i.e., $\sigma_w = 0.04$). Figure 4(e) and (f) shows the effect of N , which varies from 20 to 100 and from 200 to 1,000, respectively, with $W = 12$ and SNR = 25. It is obvious from these figures that the SD estimated from (15) has the lowest ARE% in all cases, except for $W < 6$ when Guo's algorithm

is the best. This is called the *accurate property* of the FAS algorithm. In many practical applications, an adequate portion of the GF (i.e., $W \geq 6$) is sampled with more than 200 observation points (i.e., $N \geq 200$). Roonizi's algorithm is more general than the other techniques since it can also fit a Gaussian riding on a polynomial background. This might explain its poorer performance in comparison to the other algorithms that fit a sole GF as described by (1).

The plots in Figure 4 also depict the worst-case ARE% of the proposed algorithm. The simulated worst-case ARE% represents the maximum ARE% that occurs during the 10^4 simulated trials, which is compared to (18) with $k_1 = 2$ and $k_2 = 3$ to show the accuracy of our derived error estimated in (18). Note that the probability of such a worst-case error is very low. Notably, the worst-case theoretical and simulated ARE% match, except when $W < 6$ due to the

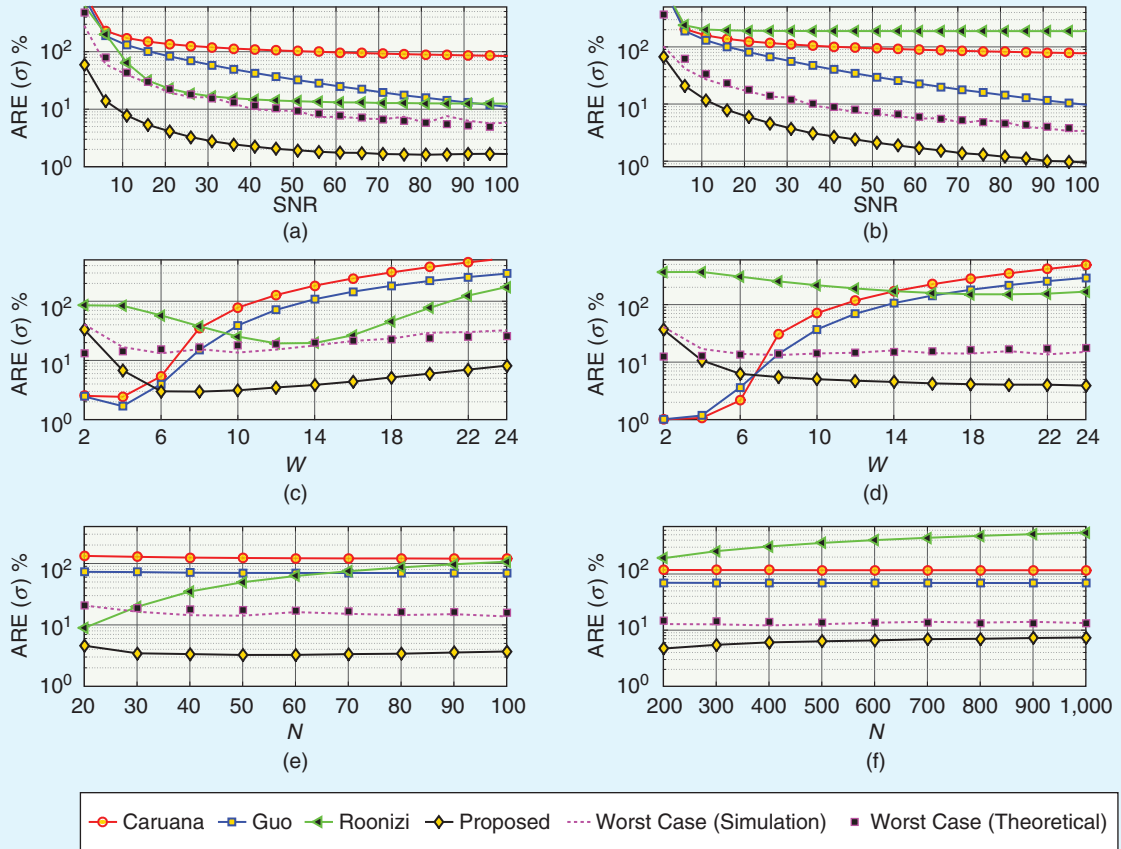


FIGURE 4. The ARE% of σ estimated from different fitting algorithms. (a) $W = 12$ and $N = 30$. (b) $W = 12$ and $N = 200$. (c) SNR = 25 and $N = 30$. (d) SNR = 25 and $N = 200$. (e) $W = 12$ and SNR = 25. (f) $W = 12$ and SNR = 25.

considerable systematic error. The superiority of the proposed algorithm versus the traditional ones holds for the worst-case ARE% as well; however, for the clarity of the plots in Figure 4, curves corresponding to the latter algorithms were not included. As shown in Figure 4(f), after a particular value of N , the error of the denominator in (15) becomes dominant. As N increases, there will be many samples around the peak of the GF, and the ARE% of the proposed algorithm slightly increases when N increases, finally approaching the worst-case scenario.

Complexity comparison

We address the computational complexity comparison of Guo's, Roonizi's, and the proposed FAS algorithms in terms of the number of additions and multiplications required to complete the fitting procedure. We assume that subtraction and division operations are respectively equivalent to addition and multiplication operations in complexity. It should be noted that solving an $n \times n$ linear system of equations using Gauss elimination requires $(2n^3 + 3n^2 - 5n)/6$ additions and $(n^3 + 3n^2 - n)/3$ multiplications [8]. Therefore, the total number of additions (Add) and multiplications (Mul) for the Guo, Roonizi, and FAS algorithms are given as follows:

$$\begin{aligned} \text{Add}^{(\text{Guo})} &= N(A_{\text{in}} + 8) + 3, \\ \text{Mul}^{(\text{Guo})} &= N(M_{\text{in}} + 11) + 17, \end{aligned} \quad (21)$$

$$\begin{aligned} \text{Add}^{(\text{Roonizi})} &= N^2 + 8N + NA_{\text{exp}} - 5, \\ \text{Mul}^{(\text{Roonizi})} &= 0.5N^2 + 9.5N + NM_{\text{exp}} + 9, \end{aligned} \quad (22)$$

$$\begin{aligned} \text{Add}^{(\text{FAS})} &= N(A_{\text{in}} + 8) - 3, \\ \text{Mul}^{(\text{FAS})} &= N(M_{\text{in}} + 10) + 12, \end{aligned} \quad (23)$$

where A_{in} and M_{in} represent the number of additions and multiplications required to calculate the natural logarithm, respectively, while A_{exp} and M_{exp} represent the number of additions and multiplications, respectively, required to calculate the natural exponential in (12). Note that the term of N^2 in (22) comes from the calculation of $\phi_1(x)$ in (9), which requires an accumulated numerical integration of $(uy(u))$ from the first observation

point to the current value of x for all N observations.

It can be seen from (21) to (23) that the proposed algorithm requires fewer additions and multiplications when compared with Guo's and Roonizi's algorithms. Assuming $A_{\text{in}} = A_{\text{exp}}$ and $M_{\text{in}} = M_{\text{exp}}$, the proposed algorithm saves six additions and $O(N)$ multiplications over Guo's algorithm, while it saves $O(N^2)$ additions and multiplications over Roonizi's algorithm. This is referred to as the *fast property* of the proposed FAS algorithm.

Conclusions

This article proposed a simple approximation expression for the SD of a GF to fit a set of noisy observed data points. This expression results from a simple mathematical trick, which is based on the equality between the area under the GF calculated numerically and based on the Q -function properties. Then, the amplitude and mean of the GF can be calculated using the weighted least-squares method. Through comprehensive simulations and mathematical analysis, it has been shown that the proposed algorithm is not only faster than Guo's and Roonizi's algorithms, but also provides better estimation accuracy when an adequate interval of the GF is sampled. Additionally, an iterative procedure is proposed, which is suitable to fit the GF when the observed data points are contaminated with substantial noise, as in the case of a long-tail GF. It has been shown by extensive computer simulations that the proposed iterative algorithm fits the GF faster than the iterative Guo's algorithm. The proposed algorithm could be useful for several applications, such as Airy disk approximation, laser transmission welding, fluorescence dispersion, and many others involving digital signal processing.

Acknowledgments

We thank Prof. Roberto Togneri, *IEEE Signal Processing Magazine's* area editor, columns and forum, and the anonymous reviewers for providing valuable suggestions to improve the manuscript. We are grateful to Prof. Balazs Bank for his assistance and insightful feed-

back during the revision of this article. The support of the Natural Sciences and Engineering Research Council of Canada through its Discovery program is also gratefully acknowledged. The work of Ertugrul Basar is supported in part by the Turkish Academy of Sciences Young Scientists Award Program.

Authors

Ibrahim Al-Nahhal (ioalnahhal@mun.ca) received his B.Sc. and M.Sc. degrees in electronics and communications engineering from Al-Azhar University and Egypt-Japan University for Science and Technology, Egypt, in 2007 and 2014, respectively. He is a Ph.D. student at Memorial University, Canada. Between 2008 and 2012, he was an engineer in industry and a teaching assistant in the Faculty of Engineering, Al-Azhar University in Cairo, Egypt. From 2014 to 2015, he was a physical-layer expert at Nokia, Belgium. He holds three patents. His research interests include designs for low-complexity receivers for emerging technologies; spatial modulation; multiple-input, multiple-output systems; and sparse code multiple access.

Octavia A. Dobre (odobre@mun.ca) received her Dipl. Ing. and Ph.D. degrees from the Polytechnic Institute of Bucharest, Romania, in 1991 and 2000, respectively. She is a professor and research chair at Memorial University, Canada. She was a visiting professor at the Massachusetts Institute of Technology and a Royal Society and Fulbright scholar. Her research interests include technologies for 5G and beyond, as well as optical and underwater communications. She has published more than 250 refereed papers in these areas. She was the editor-in-chief of *IEEE Communications Letters*. She has been a senior editor and an editor with prestigious journals as well as general chair and technical cochair of flagship conferences in her area of expertise. She is a Distinguished Lecturer of the IEEE Communications Society and a fellow of the Engineering Institute of Canada.

Ertugrul Basar (ebasar@ku.edu.tr) received his B.Sc. degree from Istanbul

University, Turkey, in 2007, and his M.S. and Ph.D. degrees from Istanbul Technical University, Turkey, in 2009 and 2013, respectively. He is currently an associate professor with the Department of Electrical and Electronics Engineering, Koç University, Istanbul, Turkey, and the director of Communications Research and Innovation Laboratory. His primary research interests include multiple-input, multiple-output systems; index modulation; waveform design; visible light communications; and signal processing for communications.

Cecilia Moloney (cmoloney@mun.ca) received her B.Sc. degree in mathematics from Memorial University, Canada, and her M.A.Sc. and Ph.D. degrees in systems design engineering from the University of Waterloo, Ontario, Canada. Since 1990, she has been a faculty member at Memorial University, where she is now a professor of electrical and computer engineering. From 2004 to 2009, she held the Natural Sciences and Engineering Research Council of

Canada/Petro-Canada chair for Women in Science and Engineering, Atlantic Region. Her research interests include nonlinear signal and image processing methods, signal representations, radar signal processing, and methods for ethics in engineering and engineering education.

Salama Ikki (sikki@lakeheadu.ca) received his B.Sc. degree from Al-Isra University, Amman, Jordan, in 1996 and his Ph.D. degree in electrical engineering from Memorial University, Canada, in 2009. He is an associate professor in the Department of Electrical Engineering, Lakehead University, Ontario, Canada. From 2009 to 2010, he was a postdoctoral researcher at the University of Waterloo, Ontario, Canada. From 2010 to 2012, he was a research assistant with the Institut national de la recherche scientifique, University of Québec, Canada. He is the author of 100 journal and conference papers and has more than 4,000 citations and an H-index of 30. His research interests include cooperative networks; multiple-input, multiple-out-

put systems; spatial modulation, and wireless sensor networks.

References

- [1] H. Liu, W. Liu, X. Zhong, B. Liu, D. Guo, and X. Wang, "Modeling of laser heat source considering light scattering during laser transmission welding," *Mater. Des.*, vol. 99, pp. 83–92, June 2016.
- [2] B. Zhang, J. Zerubia, and J.-C. Olivo-Marin, "Gaussian approximations of fluorescence microscope point-spread function models," *Appl. Opt.*, vol. 46, no. 10, pp. 1819–1829, Apr. 2007.
- [3] F. Lampariello, G. Sebastiani, E. Cordelli, and M. Spano, "Comparison of Gaussian and t-distribution densities for modeling fluorescence dispersion in flow cytometric DNA histograms," *Cytometry*, vol. 12, no. 4, pp. 343–349, Jan. 1991.
- [4] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed. New York: Cambridge Univ. Press, 2007.
- [5] R. Caruana, R. Searle, T. Heller, and S. Shupack, "Fast algorithm for the resolution of spectra," *Anal. Chem.*, vol. 58, no. 6, pp. 1162–1167, May 1986.
- [6] H. Guo, "A simple algorithm for fitting a Gaussian function," *IEEE Signal Process. Mag.*, vol. 28, no. 5, pp. 134–137, Sept. 2011.
- [7] E. K. Roonizi, "A new algorithm for fitting a Gaussian function riding on the polynomial background," *IEEE Signal Process. Lett.*, vol. 20, no. 11, pp. 1062–1065, Sept. 2013.
- [8] G. Strang, *Introduction to Linear Algebra*, 3rd ed. Cambridge, MA: Wellesley-Cambridge Press, 1993.

SP

APPLICATIONS CORNER (continued from page 132)

seismic events from Llaima volcano (Chile)," in *Proc. 2018 Int. Joint Conf. Neural Networks*, pp. 1–8, doi: 10.1109/IJCNN.2018.8489285.

[45] R. Soto, F. Huenupan, P. Meza, M. Curilem, and L. Franco, "Spectro-temporal features applied to the automatic classification of volcanic seismic events," *J. Volcanology Geothermal Res.*, vol. 358, pp. 194–206, June 2018. doi: 10.1016/j.jvolgeores.2018.04.025.

[46] M. Curilem, F. Huenupan, D. Beltrán, C. San Martín, G. Fuentealba, L. Franco, C. Cardona, G. Acuña et al., "Pattern recognition applied to seismic signals of Llaima volcano (Chile): An evaluation of station-dependent classifiers," *J. Volcanology Geothermal Res.*, vol. 315, pp. 15–27, Apr. 2016. doi: 10.1016/j.jvolgeores.2016.02.006.

[47] S. M. Bhatti, M. S. Khan, J. Wuth, F. Huenupan, M. Curilem, L. Franco, and N. B. Yoma, "Automatic detection of volcano-seismic events by modeling state and event duration in hidden Markov models," *J. Volcanology Geothermal Res.*, vol. 324, pp. 134–143, Sept. 2016. doi: 10.1016/j.jvolgeores.2016.05.015.

[48] D. A. Firoozabadi, F. Seguel, I. Soto, D. Guevara, F. Huenupan, M. Curilem, and L. Franco, "Evaluation of Llaima volcano activities for localization and classification of LP, VT and TR events," *J. Electr. Eng.*, vol. 68, no. 5, pp. 325–338, 2017. doi: 10.1515/jee-2017-0064.

[49] J. C. Lahr, B. A. Chouet, C. D. Stephens, J. A. Power, and R. A. Page, "Earthquake classification, location, and error analysis in a volcanic environment: Implications for the magmatic system of the 1989–1990 eruptions at redoubt volcano, Alaska," *J. Volcanology Geothermal Res.*, vol. 62, no. 1–4, pp. 137–151, 1994. doi: 10.1016/0377-0273(94)90031-0.

[50] C. Bouvet de Maisonneuve, M. A. Dungan, O. Bachmann, and A. Burgisser, "Insights into shallow magma storage and crystallization at Volcán Llaima (Andean Southern Volcanic Zone, Chile)," *J. Volcanology Geothermal Res.*, vol. 211–212, pp. 76–91, Jan. 2012. doi: 10.1016/j.jvolgeores.2011.09.010.

[51] L. E. Franco, J. L. Palma, F. Gil-Cruz, and J. J. San Martín, "Descripción de la actividad sísmica relacionada a erupciones estrombolianas violentas: Vn. Llaima, Chile (2007–2010)," *Earth Sci. Res. J.*, vol. 18, pp. 338–339, July 2014.

[52] M. Curilem, R. F. de Mello, F. Huenupan, C. San Martín, L. Franco, E. Hernández, and R. A. Rios, "Discriminating seismic events of the Llaima volcano (Chile) based on spectrogram cross-correlations," *J. Volcanology Geotherm. Res.*, vol. 367, pp. 63–78, Nov. 2018. doi: 10.1016/j.jvolgeores.2018.10.023.

[53] M. Titos, A. Bueno, L. García, M. C. Benítez, and J. Ibañez, "Detection and classification of continuous volcano-seismic signals with recurrent neural networks," *IEEE Trans. Geosci. Remote Sens.*,

vol. 57, no. 4, pp. 1936–1948, 2019. doi: 10.1109/TGRS.2018.2870202.

[54] M. Malfante, M. Dalla Mura, J. I. Mars, J.-P. Métaixian, O. Macedo, and A. Inza, "Automatic classification of volcano seismic signatures," *J. Geophys. Res.: Solid Earth*, vol. 123, no. 12, pp. 10,645–10,658, 2018. doi: 10.1029/2018JB015470.

[55] C. Hibert, F. Provost, J.-P. Malet, A. Maggi, A. Stumpf, and V. Ferrazzini, "Automatic identification of rockfalls and volcano-tectonic earthquakes at the Piton de la Fournaise volcano using a random forest algorithm," *J. Volcanology Geothermal Res.*, vol. 340, pp. 130–142, June 2017. doi: 10.1016/j.jvolgeores.2017.04.015.

[56] L. García, I. Álvarez, M. Titos, A. Díaz-Moreno, M. C. Benítez, and Á. de la Torre, "Automatic detection of long period events based on subband-envelope processing," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 11, pp. 5134–5142, 2017. doi: 10.1109/JSTARS.2017.2739690.

[57] U. Stańczyk, B. Zielosko, and L. C. Jain, Eds., *Advances in Feature Selection for Data and Pattern Recognition*. Cham, Switzerland: Springer, 2018.

[58] L. Yu and H. Liu, "Efficient feature selection via analysis of relevance and redundancy," *J. Mach. Learning Res.*, vol. 5, pp. 1205–1224, Dec. 2004.

SP



06 – 10 July 2020

London 

General Chairs

Marta Mrak, BBC, UK
Ebroul Izquierdo, Queen Mary University of London, UK
Vladan Velisavljevic, University of Bedfordshire, UK

Program Chairs

Shuai Wan, Northwestern Polytechnical Uni., China
Ce Zhu, Uni. of Elec. Sci. & Tech. of China, China
Jian Zhang, Uni. of Technology, Sydney, Australia
Jianquan Liu, NEC Corporation, Japan
Shiwen Mao, Auburn University, US
Wen-Huang Cheng, National Chiao Tung Uni, Taiwan
Jiwen Lu, Tsinghua University, China

Plenary Chairs

Mei-Ling Shyu, University of Miami, US
Joao Ascenso, Instituto Superior Tecnico, Portugal

Workshop Chairs

Noel O'Connor, Dublin City University, Ireland
Raouf Hamzaoui, De Montfort University, UK

Special Session Chairs

Athanasios Mouchtaris, Amazon, US
Gene Cheung, York University, Canada

Tutorial Chairs

Pascal Frossard, EPFL, Switzerland
Joao Magalhaes, Uni. Nova de Lisboa, Portugal

Panel Chairs

Chia-Wen Lin, National Tsing Hua University, Taiwan
Nikolaos Boulgouris, Brunel University, UK

Industry Chairs

Vanessa Testoni, Samsung, Brazil
Shenglan Huang, Hulu, China
Béatrice Pesquet, Thales, France

Demo Chairs

Christian Timmerer, Alpen-Adria-Uni. Klagenfurt, Austria
Saverio Blasi, BBC, UK

Expo Chairs

Sebastiaan Van Leuven, Twitter, UK
Balu Adsumilli, YouTube, US

Finance Chair

Qianni Zhang, Queen Mary University of London, UK

Publication Chair

Eduardo Peixoto, University of Brasilia, Brazil

Publicity Chairs

Fiona Rivera, BBC, UK
Enrico Magli, Politecnico di Torino, Italy
Homer Chen, National Taiwan University, Taiwan
Joern Ostermann, Leibniz Uni. Hannover, Germany

Registration and Local Chairs

Tomas Piatrik, Queen Mary University of London, UK
Charith Abhayaratne, University of Sheffield, UK

Grand Challenge Chairs

Dave Bull, University of Bristol, UK
Patrick Le Callet, University of Nantes, France

European Research Projects Chair

Alberto Rabbachin, European Commission, Belgium

Award Chair

Maria Martini, Kingston University, UK

Student Program Chair

Hantao Liu, Cardiff University, UK



21st IEEE International Conference on Multimedia and Expo

Call for Papers and Proposals

The IEEE International Conference on Multimedia & Expo (ICME) has been the flagship multimedia conference sponsored by four IEEE societies since 2000. It aims at promoting exchange of the latest advances in multimedia technologies, systems, and applications from both the research and development perspectives. ICME attracts well over 1000 submissions each year, serving as the prime forum for the dissemination of knowledge in the multimedia field.

ICME 2020 again convenes leading researchers and practitioners to share the latest developments and advances in the discipline. The conference will showcase high quality oral and poster presentations, as well as relevant Workshops sponsored by IEEE societies. An exposition of multimedia products, animations and industries will be also held in conjunction with the conference. Moreover, in ICME 2020 exceptional papers and contributors will be selected and recognised with prestigious awards.

Topics of interest include, but are not limited to:

- Artificial intelligence for multimedia
- Multimedia and vision
- Development of multimedia standards and related research
- Media quality metrics and enhancements
- Multimedia communications, networking and mobility
- 3D multimedia and AR/VR
- Multimedia security, privacy and forensics
- Multimedia software, hardware and application systems
- Multimedia and language
- Multimedia analytics, search and recommendation
- Social and cloud-based multimedia
- Media classification and segmentation

Important Dates

- | | |
|----------------------------------------------------------|-------------------------|
| • Regular paper submission (including special sessions): | 29 November 2019 |
| • Submission of papers to workshops and other tracks: | 13 March 2020 |
| • Special session proposals: | 01 October 2019 |
| • Panel proposals: | 04 October 2019 |
| • Grand challenge proposals: | 11 October 2019 |
| • Workshop proposals: | 18 October 2019 |
| • Tutorial proposals: | 01 November 2019 |
| • Hands-on demo proposals: | 01 May 2020 |



2020.ieeeicme.org

 @icme2020

Timeline

Special Session Proposals

Submission:
1 Oct 2019

Panel Proposals

Submission:
4 Oct 2019

Grand Challenge Proposals

Submission:
11 Oct 2019

Workshop Proposals

Submission:
18 Oct 2019

Tutorial Proposals

Submission:
1 Nov 2019

Regular Papers

Special Session Papers

Submission: 29 Nov 2019

Workshop Papers

Grand Challenge Papers

Demo Papers

Industry Papers

Project Papers

Submission: 13 March 2020

See 2020.ieeeicme.org for more details.

CONFERENCE AND JOURNAL PAPERS

All papers presented at ICME 2020 will be included in IEEE Xplore.

The IEEE Transactions on Multimedia is partnering with IEEE ICME. Extended versions of the top-ranked ICME 2020 papers will be invited for submission and potential publication in the IEEE Transactions on Multimedia. We invite authors to [submit high-quality contributions](#) aiming at taking part in this call and to have the opportunity to publish an extended ICME paper in this prestigious journal. After the due review process, if your paper is highly ranked, the Technical Program Committee Chairs will get in touch with you regarding the next steps.

Hosts



ICME 2020 will be hosted by Queen Mary University of London (QMUL). QMUL is one of the biggest University of London colleges and one of 24 leading UK universities represented by the Russell Group, that are committed to maintaining the very best research, an outstanding teaching and learning experience, excellent graduate employability and unrivalled links with business and the public sector.

The Multimedia and Vision Research Group (MMV) at QMUL conduct research in image, video processing, computer vision and machine learning, and are recognised as a key-player in multimedia computing in the EU and beyond. The strategic objectives of the group are to address critical technical challenges in emerging multimedia applications by strengthening existing research fields and building new research initiatives in a dynamic way according to current trends in technology; to enhance outreach activities by working closer with the industry and other research labs world-wide.



Please send calendar submissions to:
Dates Ahead, Attn: Samantha Walter, E-mail: walter.samantha@ieee.org

2019

NOVEMBER

Asilomar Conference on Signals, Systems, and Computers

3–6 November, Pacific Grove, California, United States.

General Chair: Gerald Matz

URL: <http://www.asilomarssc.org>

IEEE Global Conference on Signal and Information Processing (GlobalSIP)

11–14 November, Ottawa, Ontario, Canada.

General Cochairs: Fabrice Labeau and Sreeraman Rajan

URL: <https://2019.ieeeglobalsip.org/>

Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)

18–21 November, Lanzhou, China.

General Cochairs: Thomas Fang Zheng, Hongzhi Yu, Jianwu Dang, Wan-Chi Siu, and Hitoshi Kiya

URL: <http://www.apsipa2019.org/>

DECEMBER

IEEE International Workshop on Information Forensics and Security (WIFS)

9–12 December, Delft, The Netherlands.

General Chair: Zekeriya Erkin

URL: <https://wifs2019.tudelft.nl/>

IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)

10–12 December, Ajman, United Arab Emirates.

General Cochairs: Khaled Assaleh and Adel Elmaghraby

URL: <http://www.isspit.org/isspit/2019/index.html>

IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)

14–18 December, Sentosa, Singapore.

General Chair: Haizhou Li

General Cochair: Eric Fosler-Lussier

URL: <http://www.asru2019.org/>

Digital Object Identifier 10.1109/MSP.2019.2920036
Date of current version: 29 October 2019



The IEEE International Conference on Image Processing will be held 25–28 October 2020 in Abu Dhabi, United Arab Emirates.

IEEE International Workshop on Computational Advances in Multisensor Adaptive Processing (CAMSAP)

14–18 December, Guadeloupe, West Indies.

General Chairs: David Brie and Jean-Yves Tourneret

URL: <https://camsap19.ig.fpmc.ac.be>

2020

APRIL

IEEE International Symposium on Biomedical Imaging (ISBI)

4–8 April, Iowa City, Iowa, United States.

General Chairs: Mathews Jacob and Jong Chul Ye

URL: <http://2020.biomedicalimaging.org/>

MAY

IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)

4–9 May, Barcelona, Spain.

General Chairs: Ana I. Pérez Neira and Xavier Mestre

URL: <https://2020.ieeeicassp.org>

JUNE

IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM)

8–11 June, Hangzhou, China.

General Chairs: Martin Haardt and Zhiguo Shi

URL: <http://www.sam2020.cn>

JULY

IEEE International Conference on Multimedia and Expo (ICME)

6–10 July, London, United Kingdom.

General Chairs: Marta Mark, Ebroul Izquierdo, and Vladan Velisavljevic

URL: <http://www.2020.ieeeicme.org/>

OCTOBER

IEEE International Conference on Image Processing (ICIP)

25–28 October, Abu Dhabi, United Arab Emirates.

General Chairs: Mohammed Al-Mualla and Moncef Gabbouj

URL: <http://2020.ieeeicip.org/>

the promising emerging signal processing technologies that may apply. For instance, with the trend toward adoption of machine-learning techniques to solve conventional signal processing applications, we expect increased industry scrutiny of how practical and implementable these new approaches can be.

If you are interested in getting involved with and/or contributing to the IDSPT-SC, please contact Mike Polley. To learn more about the IDSPT-SC, please visit <https://signalprocessing.society.org/get-involved/industry-dsp-technology-standing-committee>.

Authors

Mike Polley (polley@alum.mit.edu) received his B.S., M.S., and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, in 1989, 1990, and 1996, respectively. He is a senior vice president and head of the Mobile Processor Innovation Lab at Samsung Research America, Plano, Texas. His current research interests include computational imaging and artificial intelligence-based camera systems. He currently serves as chair of the Industry Digital Signal Processing Technology Standing Committee for 2019–2020. He is a Senior Member of the IEEE.

Fa-Long Luo (f.luo@ieee.org) received his B.S., M.S., and Ph.D. degrees from Xidian University, Xi'an, China, in 1983, 1989, and 1992, respectively. He is a chief scientist at Micron Technology, San Jose, California. He has published six books and received 61 patents/inventions in signal processing, wireless communications, computing platform, and memory access. He served as the chair of the Industry Digital Signal Processing Technology Standing Committee from 2011 to 2012 and 2017 to 2018. He is a Fellow of the IEEE.

SP

IEEE connects you to a universe of information!

As the world's largest professional association dedicated to advancing technological innovation and excellence for the benefit of humanity, the IEEE and its Members inspire a global community through its highly cited publications, conferences, technology standards, and professional and educational activities.

Visit www.ieee.org.

Publications / IEEE Xplore® / Standards / Membership / Conferences / Education



IEEE

IMAGE LICENSED BY INGRAM PUBLISHING

The Industry Digital Signal Processing Technology Standing Committee

The IEEE Industry Digital Signal Processing Technology Standing Committee (IDSPT-SC) is one of 13 TCs in the IEEE Signal Processing Society (SPS). Our mission is to serve as a unique technical bridge, connecting academia and industry with people in all areas related to signal processing. We aim to promote both industry participation in the SPS and other IEEE Societies as well as industry adoption of cutting-edge signal processing technology. Signal processing areas covered by the committee span those of other TCs in the SPS, but with increased emphasis on practical industry applications of signal processing. At this time, we have 19 active members, including 13 from leading industry organizations and six from highly recognized universities. Currently, both the chair and vice chair (past chair) of the committee are from industry. The high percentage of industry participation increases the focus on immediately important applications, commercial-ready implementations, and key signal processing technologies that are ready for industry adoption.

The IDSPT-SC organizes and manages the Industry Technology Track at ICASSP and covers varied interests with 10 broad categories and 36 subcategories, as shown in the SPS Editors Information Classification Scheme. Member responsibilities include reviewing and selecting papers, organizing and chairing sessions, and nominating best

paper award candidates. Beyond handling the core Industry Technology Track, we founded the well-received interactive show-and-tell demo sessions for both ICASSP and ICIP and also organized industry panel sessions for ICASSP.

In addition to ICASSP contributions, the IDSPT-SC organizes industry-related special issues for *IEEE Transactions on Signal Processing* and *IEEE Signal Processing Magazine*. We also organize and staff workshops related to the applications of signal/image processing technologies in industry, e.g., the Emerging Signal Processing

Applications symposium at GlobalSIP. Additionally, we frequently nominate deserving papers for IEEE SPS Paper Awards and nominate spectacular individuals for IEEE SPS Major Awards. The IDSPT-SC also implemented services and activities such as signal processing-related standard developments and professional certification programs as well as promoted continuing education of the SPS membership's industrial sector.

Simply put, the IDSPT-SC provides a forum for industry people and academics to come together and share current, industry-focused interest areas and



The past, present, and nominated future chairs of IDSPT-SC at ICASSP 2019 in Brighton, United Kingdom (from right): Fa-Long Luo, Mike Polley, and Ivan Tashev.

Call for Papers
IEEE Journal of Selected Topics in Signal Processing
Special Issue on Domain Enriched Learning for Medical Imaging

In recent years, learning based methods have emerged to complement traditional model and feature based methods for a variety of medical imaging problems such as image formation, classification and segmentation, quality enhancement etc. In the case of deep neural networks, many solutions have achieved unprecedented performance gains and have defined a new state of the art. Despite the progress, compelling open challenges remain. One such key challenge is that many learning frameworks (notably deep learning) are purely data-driven approaches and their performance depends strongly on the quantity and quality of training image data available. When training is limited or noisy, the performance drops sharply. Deep neural networks based approaches additionally face the challenge of often not being straightforward to interpret. Fortunately, exciting recent progress has emerged in enriching learning frameworks with domain knowledge and signal structure. As a couple of representative examples: in image reconstruction problems, this may involve using statistical/structural image priors; for image segmentation, shape and anatomical knowledge (conveyed by an expert) may be leveraged, *etc.* This special issue invites original new contributions that combine signal, image priors and other flavors of domain knowledge with machine learning methods for solving medical imaging problems. Topics of interest include but are not limited to:

- Fundamental innovations in combining model based and learning based methods.
- Sparse representation and dictionary learning based methods for medical image processing and understanding.
- Domain enriched and regularized deep learning via special network architectures and systematic integration of problem specific insights.
- Interpretable deep networks for medical imaging via techniques such as algorithm unrolling.
- Algorithmic methods that gracefully degrade with amount of training image data available and enable robustness against selection bias.
- Example applications include image reconstruction and formation, medical image classification and segmentation, image understanding, boundary and shape analysis, registration, quality enhancement etc. The scope encompasses all medical imaging modalities including but not limited to MRI, X-Ray, CT, PET, ultrasound, photoacoustic imaging, various forms of microscopy, multispectral imaging, new and emerging imaging techniques and modalities.

Important Dates:

| | |
|----------------------------|------------------------------|
| Manuscript Submission Due: | Dec. 01 st , 2019 |
| First Review Completed: | Feb. 01 st , 2020 |
| Revised Manuscript Due: | Apr. 01 st , 2020 |
| Second Review Completed: | Jun. 01 st , 2020 |
| Final Manuscript Due: | Jul. 15 th , 2020 |

Guest Editors (GEs):

Vishal Monga (Lead GE, Pennsylvania State University, USA, vmonga@engr.psu.edu)

Scott Acton (University of Virginia, USA, acton@virginia.edu)

Arrate Muñoz Barrutia (Universidad Carlos III de Madrid, Spain, amunoz@hggm.es)

Jong Chul Ye (KAIST - Korea Advanced Institute of Science and Technology, jong.ye@kaist.ac.kr)

Abd-Krim Seghouane (University of Melbourne, Australia, abd-krim@seghouane@unimelb.edu.au)

mathworks.com/deeplearning

