Learning-based Visual Compression

Other titles in Foundations and Trends® in Computer Graphics and Vision

Video Summarization Overview
Mayu Otani, Yale Song and Yang Wang
ISBN: 978-1-63828-078-1

A Comprehensive Review of Modern Object Segmentation Approaches
Yuanbo Wang, Unaiza Ahsan, Hanyan Li and Matthew Hagen
ISBN: 978-1-63828-070-5

Deep Learning for Image/Video Restoration and Super-resolution
A. Murat Tekalp
ISBN: 978-1-68083-972-2

Deep Learning for Multimedia Forensics
Irene Amerini, Aris Anagnostopoulos, Luca Maiano and Lorenzo Ricciardi Celsi
ISBN: 978-1-68083-854-1

Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art
Joel Janai, Fatma Güney, Aseem Behl and Andreas Geiger
ISBN: 978-1-68083-688-2

Discrete Graphical Models - An Optimization Perspective
Bogdan Savchynskyy
ISBN: 978-1-68083-638-7

Learning-based Visual Compression

Ruolei Ji
Arizona State University
ruoleiji@asu.edu

Lina J. Karam
Lebanese American University
Arizona State University
lina.karam@lau.edu.lb
karam@asu.edu

now
the essence of knowledge
Boston — Delft

Foundations and Trends® in Computer Graphics and Vision

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

R. Ji and L. J. Karam. *Learning-based Visual Compression*. Foundations and Trends® in Computer Graphics and Vision, vol. 15, no. 1, pp. 1–112, 2022.

ISBN: 978-1-63828-113-9

Foundations and Trends® in Computer Graphics and Vision

Volume 15, Issue 1, 2022

Editorial Board

Editorial Scope

Topics

Foundations and Trends® in Computer Graphics and Vision publishes survey and tutorial articles in the following topics:

- Rendering
- Shape
- Mesh simplification
- Animation
- Sensors and sensing
- Image restoration and enhancement
- Segmentation and grouping
- Feature detection and selection
- Color processing
- Texture analysis and synthesis
- Illumination and reflectance modeling
- Shape representation
- Tracking
- Calibration
- Structure from motion
- Motion estimation and registration
- Stereo matching and reconstruction
- 3D reconstruction and image-based modeling
- Learning and statistical methods
- Appearance-based matching
- Object and scene recognition
- Face detection and recognition
- Activity and gesture recognition
- Image and video retrieval
- Video analysis and event recognition
- Medical image analysis
- Robot localization and navigation

Information for Librarians

Foundations and Trends® in Computer Graphics and Vision, 2022, Volume 15, 4 issues. ISSN paper version 1572-2740. ISSN online version 1572-2759. Also available as a combined paper and online subscription.

Contents

Learning-based Visual Compression

Ruolei Ji[1] and Lina J. Karam[1,2]

[1] *Arizona State University, USA; ruoleiji@asu.edu, karam@asu.edu*
[2] *Lebanese American University, Lebanon; lina.karam@lau.edu.lb*

ABSTRACT

Visual compression is an application of data compression to lower the storage and/or transmission requirements for digital images and videos. Due to the rapid growth in visual data transmission demand, more efficient compression algorithms are needed. Considering that deep learning techniques have successfully revolutionized many visual tasks, learning-based compression algorithms have been explored over the years and have been shown to be able to outperform many conventional compression methods. This survey provides a review of various visual compression algorithms, both end-to-end learning-based image compression approaches and hybrid image compression approaches. Some learning-based video compression methods are also discussed. In addition to describing a wide range of learning-based image compression approaches that have been developed in recent years, the survey describes widely used datasets, presents recent standardization efforts, and discusses potential research directions.

Ruolei Ji and Lina J. Karam (2022), "Learning-based Visual Compression", Foundations and Trends® in Computer Graphics and Vision: Vol. 15, No. 1, pp 1–112. DOI: 10.1561/0600000101.

1

Introduction

In recent years, the demand for visual media has been growing exponentially. According to the 2019 CISCO Visual Network Index (VNI) forecast update (Nowell, 2019), the global IP video traffic will be 82% of all IP traffic in 2022, up from 73% in 2016 (CISCO, 2016). Among the large amount of visual traffic over the Internet, high-resolution visual content constitutes an increasingly large percentage. Despite the increase in the average broadband speed, about 1.9-fold from 2016 to 2021, the growth rate of visual data, approximately 3-fold from 2016 to 2021, is much higher than the broadband growth rate (CISCO, 2016; Nowell, 2019). With such a rapid growth of digital visual media traffic, there is a growing need for image/video compression approaches that can achieve much higher compression ratios than the ones obtained using existing conventional image/video compression methods, while maintaining a high visual quality.

Most conventional lossy image compression methods, *e.g.,* JPEG (Wallace, 1992), WebP (Google, 2015), JPEG2000 (Taubman and Marcellin, 2002), BPG[1] (Bellard, 2018), HEVC-based intra coding (Sullivan

[1]This corresponds to a container for HEVC-based intra coding, Main 4:4:4 16 intra profile.

et al., 2012), and VVC-based intra coding (Ohm and Sullivan, 2018) are built based on a transform coding based framework (Goyal, 2001), where an invertible transform module is used to map image pixel intensities or predicted pixel residuals into a latent representation at the encoder. The latent representation is then quantized to produce a compact representation. An entropy encoder is later employed for coding the quantized latent representation. At the decoder, an inverse transform module is applied to the entropy decoded quantized data to recover a lossy image.

Although the conventional compression algorithms have been widely used and achieved promising results, researchers are working on using learning-based approaches for image/video compression to help further improve the compression performance. Such increased interest in learning-based compression stems from the fact that, over the last decade, deep-learning-based approaches have achieved huge success in a variety of visual tasks including but not limited to classification, segmentation, object detection, and super-resolution.

2

Learning-based Visual Compression Methods

Both conventional compression algorithms and learning-based compression algorithms follow the transform-based framework (Goyal, 2001), as shown in Figure 2.1, in which a transform, which is hand-designed for conventional compression methods and trainable for learning-based compression approaches, is employed to map the image pixel intensities into a latent representation. For lossy image compression, the latent representation is further quantized. The resulting (quantized) latent representation is losslessly entropy-encoded in order to generate a bitstream for transmission and/or storage. At the decoder side, the entropy-encoded (quantized) latent representation is entropy-decoded from the bitstream and fed into an inverse transform module to get the (lossy) reconstructed image.

Several learning-based compression methods have been developed over the years. These learning-based compression methods can be divided into two groups based on whether or not a conventional codec is leveraged. Methods in which a conventional codec with a hand-crafted transform is leveraged are referred to as "hybrid learning-based compression methods." Methods in which both the transform and inverse transform are learned are referred to as "end-to-end learning-based compression methods."

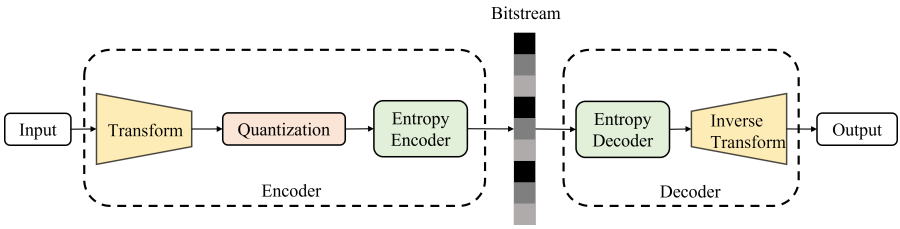

Figure 2.1: Illustration of a transform-based compression framework. The quantization module is not present for lossless compression.

Most end-to-end learning-based compression methods are autoencoder-based approaches, in which a learning-based transform, usually implemented through Convolutional Neural Networks (CNNs), is used in lieu of the traditional hand-crafted transform (e.g., Discrete Cosine Transform) in a transform coding framework. In addition to autoencoder-based approaches, "generative compression" methods, in which a GAN-based framework is used in conjunction with a CNN-based compression system, have been explored in recent years (Rippel and Bourdev, 2017; Agustsson *et al.*, 2019; Kudo *et al.*, 2019).

2.1 Hybrid Learning-based Compression Methods

Hybrid learning-based compression methods are methods in which a conventional compression method is used in conjunction with learning-based pre- and/or post-processing modules. Based on how the learning-based methods are utilized in conventional compression methods, these hybrid learning-based methods can be further divided into three categories: a) decoder-side post-processing, b) encoder-side pre-processing, and c) Deep Neural Network (DNN)-based modules used in lieu of some select portions of the conventional compression method.

2.1.1 Decoder-Side Post-Processing

The decoder-side post-processing consists of adding a learning-based module right after the decoder. As illustrated in Figure 2.2, the post-processing module consists of Deep Neural Network (DNN) based quality enhancement networks that are applied to the output image that re-

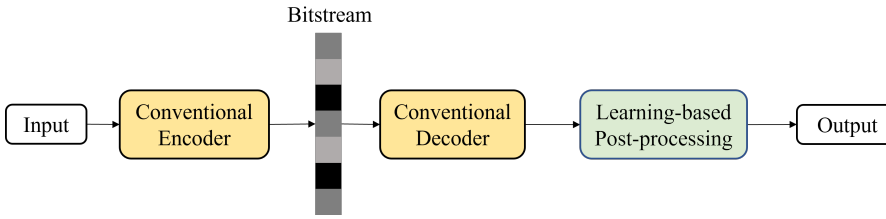

Figure 2.2: Block diagram illustrating a conventional compression system with a learning-based decoder-side post-processing.

sulted from the employed conventional lossy compression/decompression system.

Lu *et al.* (2019b) proposed a learning-based image restoration network as a decoder-side post-processing module to further improve the visual quality of the reconstructed RGB image that is obtained using a Versatile Video Coding (VCC) Intra Profile based image codec (VVC Intra). Since VCC Intra only takes YUV instead of RGB as input, the authors use FFmpeg to convert the RGB input image to a VCC-compliant YUV image. A corresponding color conversion is then used to transform the decoded YUV image back to RGB format. The proposed restoration network is applied to the reconstructed RGB image and, by capturing multi-scale spatial priors, the network is able to improve the visual quality of the reconstructed RGB image. Based on a Super-Resolution Convolutional Neural Network (SR-CNN) (Dong *et al.*, 2014), which is a deep CNN used to perform single-image super-resolution, Dong *et al.* (2015) proposed the Artefacts Reduction (AR)-CNN, consisting of a 4-layer fully convolutional neural network, to reduce visible artefacts that are generated by the lossy compression. In the work of Dong *et al.* (2015), the post-processing module is trained for the standard JPEG compression scheme.

Such decoder-side post-processing strategy can also be applied to video compression methods. Wang *et al.* (2017) proposed the Deep CNN-based Auto Decoder (DCAD), which is a 10-layer CNN, to automatically remove artefacts and enhance the visual quality of videos that are compressed/decompressed using the High Efficiency Video Codec (HEVC) (Sullivan *et al.*, 2012). In DCAD, rather than directly

predicting a higher quality reconstructed frame, the networks predict the residual and this residual is later added to the input. Different from DCAD in which the same CNN model is used to enhance both intra- and inter-coding frames, Yang *et al.* (2019) proposed a novel Quality Enhancement CNN (QE-CNN) for enhancing the decoded video of HEVC more efficiently. QE-CNN uses different network architectures for HEVC intra- and inter-coding, QE-CNN-I for HEVC I frames and QE-CNN-P for HEVC P/B frames.

Other than enhancing the decompressed images through artefact reduction, Generative Adversarial Networks (GANs) have been used to improve the perceived visual quality of the decompressed images. Fatima *et al.* (2021) proposed a framework in which a GAN-based image colorization is employed to help improve the compression performance. The raw RGB color image is converted to the YUV color space and only the luminance channel (Y) is compressed and decompressed using a conventional compression method (JPEG is used in this work). Meanwhile, a sparse color seed is transmitted from the encoder side to the decoder side. This color seed, along with the decoded lossy luminance channel, are fed to the GAN-based colorization network to generate the colorized reconstructed image. The GAN-based colorization network used in Fatima *et al.* (2021) is adopted from Zhang *et al.* (2017). Zhang *et al.* (2017) proposed two approaches for colorization: user-guided colorization in which few pixels are chosen and assigned desired colors, and data-driven automatic colorization in which the network is trained on a large dataset and can perform the colorization automatically. In Fatima *et al.* (2021) if no seeds are transmitted, the automatic colorization mode of the GAN-based colorization system is used.

2.1.2 Encoder-Side Pre-Processing

The encoder-side pre-processing consists of adding a learning-based module before the encoder. Such encoder-side pre-processing could be paired with a decoder-side post-processing module. Figure 2.3 illustrates a conventional compression system together with an encoder-side pre-processing module.

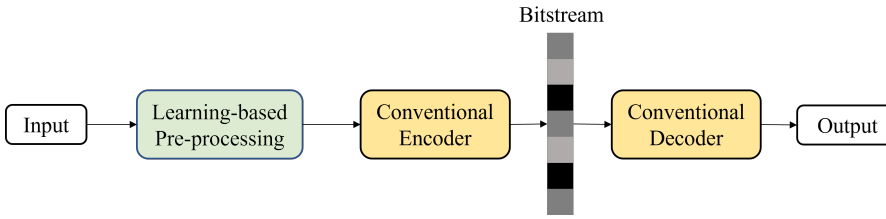

Figure 2.3: Block diagram illustrating a conventional compression system with a learning-based encoder-side pre-processing.

Many popular lossy image compression methods encode images in YUV color format while the images are originally acquired in an RGB format. Thus one pre-processing module is the learned color space conversion from RGB to YUV, which is used in lieu of the conventional RGB-to-YUV conversion that is performed by means of a matrix multiplication operation. Instead of using a fixed conversion matrix, Li (2019) proposed a learning-based color space conversion (ABC) for H.266 (Versatile Video Coding Test Model, VTM 4.0). At the encoder side, the ABC algorithm employs the Principal Component Analysis (PCA) technique to generate an RGB-to-YUV conversion matrix that is adapted for each image. Data samples for the PCA are the pixels' color in RGB minus corresponding averages. The averages are computed by taking the average of each color (R, G, B) channel over the whole image or in a locally adaptive manner by computing each channel average over a 16×16 pixel grid (Li, 2019).

At the decoder-side, the least square method (LSM) is adopted to estimate the inverse conversion matrix. Results show that the 16×16 grid's average outperforms the average over the whole image and that the LSM inverse conversion can help to further improve the visual quality (measured using the RGB-PSNR quality metric) of the reconstructed image. Later, Li *et al.* (2019) proposed a framework called VimicroABC-net which combines the ABC algorithm (Li, 2019) with a post enhancing network that is added at the decoder side. In VimicroABCnet, the ABC algorithm is implemented with a 64×64 convolutional filter kernel instead of the original 16×16 grid's average. At the decoder side, the decoded YUV output is first transformed to RGB with the LSM-learned

inverse conversion matrix. Then a CNN-based post-processing enhancement network borrowed from Zhou *et al.* (2018) is used to further help improve the reconstructed RGB image quality.

Based on VimicroABCnet, Sun *et al.* (2020) proposed a coding framework named VIP-ICT codec to improve the compression rate of the Versatile Video Coding Test Model (VTM) and the visual quality of the reconstructed image. In the VIP-ICT codec, a PCA-like algorithm similar to the ABC algorithm is used to convert the RGB input image to YUV format. Different from VimicroABCnet, the learning-based post-processing enhancement filter consisting of 11 convolutional layers, is directly learned in the YUV space to learn the YUV residual. A linear regression is used to map the learned YUV residual to the distortion between the original YUV image and the VTM reconstruction and results in a mapped YUV reconstruction by adding this distortion to the VTM reconstruction. The mapped YUV reconstruction is converted to the RGB format using the LSM-learned inverse conversion matrix.

The aforementioned learning-based compression methods mainly focus on the color format conversion. There are also several learning-based compression methods in which a pair of pre- and post-processing modules are used and the whole compression network is trained end-to-end with the conventional image compression method working as a fixed module. Tao *et al.* (2017) proposed a pair of pre-processing and post-processing modules corresponding to a de-resolution module and a super-resolution module, named as CrCNN and ReCNN, and working at the encoder side and decoder side, respectively. The input image is subjected to a de-resolution operation to obtain a compact representation through the CrCNN module and is then encoded, transmitted and decoded using conventional image codecs, such as JPEG, JPEG2000 and BPG. The restored low-resolution image is later super-resolved using the ReCNN to obtain the final reconstructed image.

Akbari *et al.* (2019) proposed a deep semantic segmentation-based layered image compression (DSSLIC) framework. In the DSSLIC codec, a pre-trained segmentation network, PSPNet (Zhao *et al.*, 2017) is used to generate the segmentation map of the input image which is encoded as the base layer of the bitstream. The segmentation map along with the input image are fed into a 5-layer convolutional neural network

followed by a Hyperbolic Tangent function, referred to as Compact Net, to obtain a low-dimensional compact representation which is encoded as the first enhancement layer. Both the segmentation map and the compact representation are used as input to a ResNet at the encoder side to obtain the reconstructed image and a residual image. The residual image is then encoded as another enhancement layer. All three layers (segmentation map, compact image, and residual image) are encoded and decoded using a lossless image codec, FLIF (Sneyers and Wuille, 2016). At the decoder side, the segmentation map and the compact image are used to generate a reconstructed image using a ResNet and later by adding the transmitted residual image, the final reconstructed image is obtained.

2.1.3 DNN-based Modules as Part of Conventional Codecs

In some approaches, deep learning based modules with deep neural networks (DNNs) are used as part of the conventional compression methods.

Figure 2.4 shows the video coding scheme which is used in HEVC (Sullivan *et al.*, 2012) and VVC (Ohm and Sullivan, 2018). The input video picture is first divided into Coding Tree Units (CTUs). Each CTU will later be further partitioned into Coding Units (CUs) to be encoded. Blockwise inter- and intra-prediction will be performed to generate the prediction residual. The prediction residual is transformed, scaled and quantized before being entropy encoded. The reconstructed signal after inverse transform, scaling and inverse quantization is then filtered by a deblocking filter, Sample Adaptive Offset (SAO) filter, Adaptive Loop Filter (ALF) and Cross-Component Adaptive Loop Filter (CC-ALF), to further improve the visual quality of the reconstructed video frames. In HEVC and VVC, an all-intra mode can be used to perform image compression. Almost all the functional modules in the video compression framework can be replaced by a DNN-based module. In this section, we will introduce some of the works for each of the functional modules. Further information on training deep networks as tools in conventional coding schemes can be found in Liu *et al.* (2020a) and Hoang and Zhou (2021).

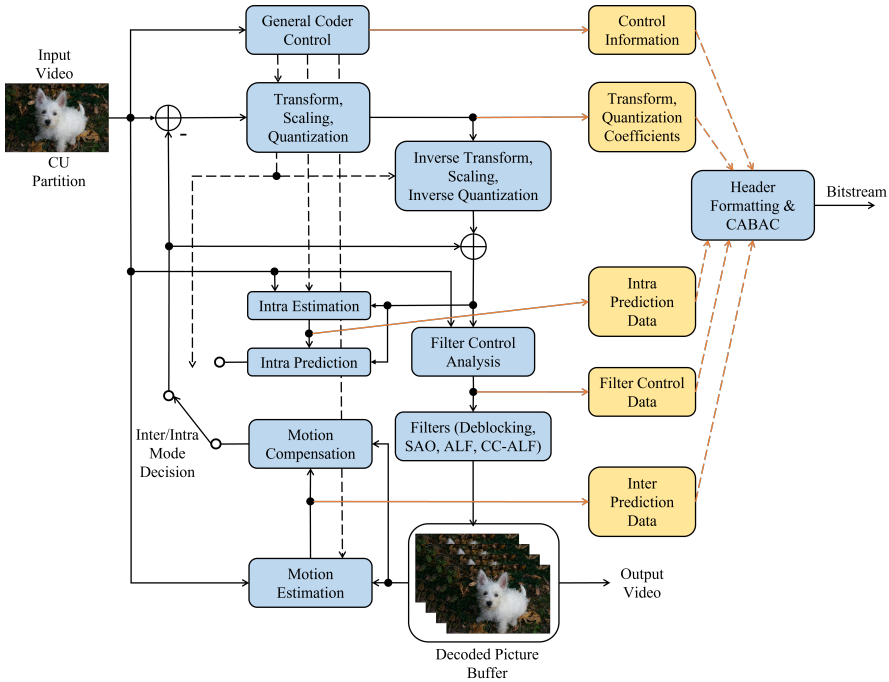

Figure 2.4: Block diagram illustrating video coding scheme for HEVC and VVC.

CU Partitioning

Compared to HEVC, the Versatile Video Coding (VVC) standard exhibits significant improvement in relation to compression performance by using a coding unit (CU) partition structure called quad-tree plus multi-type tree (QTMT) structure at the expense of a relatively large encoding time due to the search as part of the recursive rate-distortion (RD) optimization. Li *et al.* (2021) proposed DeepQTMT to predict the QTMT-based CU partition for intra-mode VVC. Li *et al.* (2021) proposed a MSE-CNN model to learn the QTMT-based CU partition. Given an input video sequence, each frame is divided into basic processing units named as Coding Tree Units (CTUs) which correspond to macroblocks in previous video coding standards. Each CTU is then partitioned into Coding Units (CUs) with different sizes before being coded. The MSE-CNN model takes a 128×128 luminance Coding Tree

Unit (CTU) as input, a convolutional layer is used to extract a group of (16) 128×128 feature maps. The extracted feature maps are used to make split decisions. Each split mode decision is fed into a conditional convolution module, consisting of several residual units, to extract textural features, which are then fed into a sub-network to predict the split mode for one CU. If the CU is decided to be non-split, the process exits, otherwise the CU will go to the next stage for further splitting. By repeating this stage several times, the CTU can have its final CU partition decision. By using the proposed MSE-CNN, Li *et al.* (2021) claim that their strategy can avoid spending significant time on checking the RD cost for all possible CUs in each CTU in the QTMT-based CU partition. Fan *et al.* (2020) and Amestoy *et al.* (2019) also explored using a learning-based approach for QTMT CU partitioning in VVC.

Intra Prediction

In the HEVC video coding standard, for intra-prediction, 35 intra-prediction modes, including one planar prediction mode for encoding planar surface, one DC mode for encoding flat surfaces and 33 angular direction modes corresponding to 33 different prediction angles, are tested during encoding to select the prediction mode with the best RD performance. Schiopu *et al.* (2019) proposed a block-wise CNN-based prediction to replace the HEVC's angular intra-prediction in order to improve the HEVC's lossless compression performance. Based on the Angular Intra-Prediction Convolutional Neural Network (AP-CNN) proposed in their previous work (Huang *et al.*, 2019), the authors specifically train a modified AP-CNN model as the prediction model for each of the HEVC angular intra-prediction modes.

Compared with HEVC, the number of intra-prediction modes is extended to 67 in the Versatile Video Coding (VVC) standard, including one planar prediction mode, one DC mode and 65 angular intra-prediction modes. Zhu *et al.* (2019) proposed a GAN-based inpainting method for intra prediction for VVC based on the work of Iizuka *et al.* (2017). The proposed intra-prediction architecture consists of two networks, a generator having 17 convolutional layers for predicting a missing block from its neighboring top-left, left and top

reconstructed blocks, and a discriminator with components operating at local and global levels. This GAN-based intra prediction network is incorporated into HEVC with 35 intra prediction modes and into VVC with 67 intra prediction modes.

Inter Prediction

Different from intra-prediction in which the prediction of a current block is performed using previous-decoded blocks within the same frame, the inter-prediction is generated by motion-compensated prediction in which the prediction is performed using previous-decoded frames. Lotter *et al.* (2016) proposed a predictive neural network (PredNet) that can predict a future frame in a video sequence based on previous frames. Based on PredNet, Benjak *et al.* (2021) proposed an enhanced learning-based inter coding algorithm for VVC. The recurrent neural network architecture of PredNet is employed to predict the picture to be coded from previous coded frames. Instead of the pixel-wise fidelity of predicted signals used in Lotter *et al.* (2016), Benjak *et al.* (2021) use the sum of absolute transformed differences (SATD) as a cost function while training PredNet. SATD is the optimization criterion used in VTM (reference software for VVC).

Video compression codecs, such as HEVC and VVC, rely on fractional-pixel motion compensation to generate motion vectors and get accurate inter-predictions. For implementation convenience, most video compression standards use fixed interpolation filters as part of the fractional-pixel motion compensation. Both HEVC and VVC use a 7-tap interpolation filter and an 8-tap interpolation filter for quarter-pixel and half-pixel motion compensation, respectively. Yan *et al.* (2018) proposed the Fractional-pixel Reference generation CNN (FRCNN) to perform fractional-pixel motion compensation in HEVC. For half-pixel interpolation, 3 FRCNN models are used, one model per pixel position, and for quarter-pixel interpolation, 15 FRCNN models are used, one model per pixel position. Each of the FRCNN models takes the reference block as input and generates the motion vector for the corresponding position. The FRCNN proposed in Yan *et al.* (2018) takes the form of an architecture consisting of N layers and where each layer, except 1 and

N, contains $k_i, 2 \leq i \leq N-1$, parallel convolutional layers followed by a concatenation operation. Layers 1 and N contain only 1 convolutional layer, to generate a residual signal for the input block.

In-loop Filter / SAO

In the HEVC video coding standard, in-loop deblocking and sample adaptive offset (SAO) filters are used to reduce the artifacts that are generated during compression. Dai *et al.* (2017) proposed a learning-based CNN, referred to as Variable-filter-size Residue-learning CNN (VRCNN), to be used as a post-processing module for HEVC in lieu of the in-loop deblocking and SAO filters. VRCNN is designed based on the 4-layer artefacts reduction network AR-CNN (Dong *et al.*, 2015). VRCNN makes use of variable-size filters since HEVC adopts variable block size transforms. One main difference between VRCNN and AR-CNN is that VRCNN is designed to learn a residue rather than the enhanced output. In Park and Kim (2016), the authors proposed a CNN-based in-loop filtering technique (IFCNN) to replace the in-loop SAO filter in HEVC. The IFCNN, designed from SR-CNN (Dong *et al.*, 2014), takes the output of the deblocking filter as input and outputs the predicted residue. This residue is then added to the deblocking filter output to obtain the final reconstructed image (video frame). Park and Kim (2016) also show that promising results can be obtained if IFCNN is used to replace both the in-loop deblocking and SAO filters.

Frame Interpolation

Wu *et al.* (2018) proposed a conditional interpolation model to generate frames between two key-frames. In their framework, every N-th frames (key frame) is compressed using compression methods of Toderici *et al.* (2017) and all frames in between are interpolated. Context features are first extracted from the two key-frames respectively and then used to interpolate the in-between frames using a motion compensated interpolation network which capture the motion difference in the interpolation frames. Furthermore, Wu *et al.* (2018) combined the motion compensated interpolation with a compressed residual information which will

capture the motion and appearance difference in the interpolated frames. The encoder, context model and the interpolation network are jointly trained.

Entropy Coding

Starting with the Advanced Video Coding (AVC) standard, also known as H.264 and MPEG4 Part 10 (Wiegand *et al.*, 2003), context-adaptive binary arithmetic coding (CABAC) (Marpe *et al.*, 2003) has been adopted for entropy coding. CABAC contains three steps: binarization, context modeling, and binary arithmetic coding. Although CABAC can significantly improve the compression efficiency compared to the context-adaptive variable length coding (CAVLC) (Marpe *et al.*, 2003), CABAC suffers from high computational complexity compared to CAVLC. Instead of manually designed binarization and context modeling, Ma *et al.* (2018) proposed a convolutional neural network-based arithmetic coding (CNNAC) scheme to estimate the probability distribution for encoding the DC coefficients of the HEVC intra prediction residual. The neural network is adopted from DenseNet (Huang *et al.*, 2017), in which each convolutional layer receives the feature maps from all preceding layers as input, with only one dense block. The DC coefficients along with the network-generated probability distribution is fed into a multi-level arithmetic codec for encoding. Later, Ma *et al.* (2019a) proposed to further expand CNNAC to code not only the DC coefficients, but also the lowest frequency AC coefficients (AC1), the second, third, fourth and fifth lowest AC coefficients (AC2, AC3, AC4, AC5), and the position of the last non-zero coefficient (LastXY). Different DenseNet architectures are designed for different coefficients, different transform unit (TU) sizes, different QPs, different channels and different scanning orders.

Bitstream Re-compression

Using a JPEG2000 encoder, Ma *et al.* (2019b) proposed a bitstream re-compression module and a CNN-based post-processing method to further improve the compression efficiency. The bitstream re-compression module, consisting of two PixelCNN-based networks (Van Oord *et al.*,

2016; Van den Oord *et al.*, 2016) and one RNN, is employed to exploit the correlation between wavelet coefficients within and across subbands. The first PixelCNN-based network takes the current subband as input to exploit the correlation within the subband while the second PixelCNN-based network takes the output from the RNN as input. The RNN takes the previous encoded subband as input to extract the long-term context. By using shortcuts after each layer between the first and second PixelCNN-based networks, the proposed bitstream re-compression is able to entropy code the wavelet coefficients of the current subband based on not only the context in the current subband but also the long-term context from previous subbands. In addition to improving the entropy coding efficiency, a CNN-based post-processing enhancement network is also employed at the decoder-side to further improve the reconstructed image quality.

2.2 Novel End-to-End Learning-based Compression Methods

Novel end-to-end learning-based compression methods are methods in which the transform and inverse transform are learned and no conventional compression methods are leveraged. Most of the end-to-end learning-based compression methods are autoencoder-based approaches, in which a trainable learning-based invertible transform, usually implemented by using CNN-based Deep Neural Networks, is employed in lieu of traditional hand-crafted transforms in a transform coding framework. The second group of end-to-end learning-based compression methods are "generative compression methods" in which a GAN-based framework is used in conjunction with a CNN-based compression system. Figure 2.5 shows the block diagram of an end-to-end learning-based compression method.

2.2.1 Autoencoder-based Approaches

Autoencoder-based approaches are developed based on the Variational Autoencoders (VAEs) (Kingma and Welling, 2013). Similar to the lossy compression methods that are based on a transform coding framework, this category of learning-based compression methods usually employ a

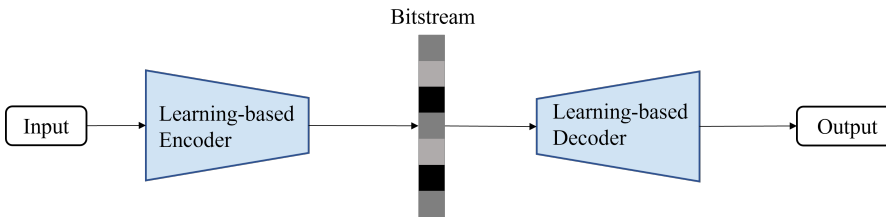

Figure 2.5: Block diagram illustrating an end-to-end learning-based compression method.

trainable neural network-based transform to map the values of the input image or of the residual image, to a compact latent representation which can be later quantized for coding. Then an entropy model incorporating a prior probability model of the discrete quantized latent representation is employed for entropy coding/decoding and is known to both the encoder and decoder. Thus, the quantized latent representation can be encoded and decoded losslessly using entropy coding algorithms such as arithmetic coding (Rissanen and Langdon, 1981).

Toderici *et al.* (2015) proposed an image compression framework that supports variable compression rates without the need for retraining or for storing multiple encoding bitstreams for the image. Using different neural network architectures, four different residual encoders are proposed in Toderici *et al.* (2015), the fully-connected residual autoencoder, the fully-connected LSTM (Long Short Term Memory neural network) residual autoencoder, the convolutional/deconvolutional residual autoencoder, and the convolutional/deconvolutional LSTM residual autoencoder. All four encoders are implemented through an iterative structure such that the autoencoder can assign a varying number of bits per image patch by varying the number of iterations. For the fully-connected residual autoencoder, in which both the encoder and decoder are composed of staked fully-connected layers, the first iteration encodes the original input and each subsequent iteration encodes the residual from the previous reconstruction levels. The encoded representations are later processed using a binarization technique, which results in a bitstream that can be directly stored and transmitted. At the decoder side, the outputs resulting from all the iterations are added to generate

the final reconstructed image. The fully-connected residual autoencoder is trained using two different approaches: weights are shared over all iteration stages, or weights are independent for each iteration stage.

For the convolutional/deconvolutional residual autoencoder, the fully-connected layers in the encoder are replaced with convolutional layers and are followed by a 1×1 convolutional layer with three filters while the fully-connected layers in the decoder are replaced with deconvolutional layers. For the fully-connected LSTM residual autoencoder, the encoder is composed of one fully-connected layer and two LSTM layers while the decoder has the reverse structure. Similar to the aforementioned residual autoencoders, for the LSTM residual encoder, the first iteration takes the original input in and from the second iteration onward, the network takes the residual from the previous stage as input. But different from the residual autoencoders, all iterations predict the input and output the residual between the prediction and the original input. The final convolutional/deconvolutional LSTM architecture is formed by combining convolutional and deconvolutional operations with LSTM. The authors use the convolutional/deconvolutional LSTM to replace the convolutional/deconvolutional layers in the convolutional/deconvolutional residual autoencoder. The authors later improved on their initial work in Toderici *et al.* (2017), where they proposed a recurrent neural network (RNN)-based encoder/decoder along with a binarizer that is used after the encoder to obtain a compact binary latent representation of the input. The output of the binarizer can be stored and/or transmitted to the decoder side.

Similar to the fully-connected and convolutonal/deconvolutional residual codecs of their previous work (Toderici *et al.*, 2015), the encoder, binarizer and decoder are implemented in an iterative manner, with the first iteration encoding the original input image and the subsequent ones encoding residuals. Toderici *et al.* (2017) also proposed an entropy coder referred to as BinaryRNN, with an architecture that is similar to the PixelRNN architecture (Van Oord *et al.*, 2016), to further compress the output of the encoder using binary codes. An improved recurrent architecture was proposed in Johnston *et al.* (2018). The authors of Johnston *et al.* (2018) made use of a Gated Recurrent Unit (GRU) neural network and proposed a method called hidden-state priming

which attempts to generate a better initial hidden-state for each GRU layer. Furthermore, Johnston *et al.* (2018) used a spatially adaptive bitrate (SABR) module to dynamically adjust the bitrate to the content of the image in a locally adaptive manner in order to increase the coding efficiency in terms of decreasing the bitrate for a desired visual quality or increasing the visual quality for a target fixed bitrate.

Different from the aforementioned iterative coding methods which encode an image into a layered representation and in which a portion of the bitstream (portion corresponding to a layer) is produced at each iteration, more recent popular end-to-end learning-based coding methods formulate the compression problem as generating a low-entropy discrete latent representation. Most of these methods follow the general nonlinear transform coding framework (Ballé *et al.*, 2016b) in which the signal-domain image intensity is mapped to a transform-domain latent presentation by means of an analysis transform at the encoder. This latent representation is then quantized and entropy encoded for storage and transmission. At the decoder, after entropy decoding, an inverse transform, also known as synthesis transform, is used to reconstruct the decoded image. This coding framework is used to learn a pair of analysis and synthesis transforms while performing a constrained rate-distortion optimization. In addition, a perceptual space transform can be applied to both the original input image and the reconstructed image; the perceptually transformed images can then be used in the computation of the distortion.

Based on this general nonlinear transform coding framework, Ballé *et al.* (2016b) proposed an end-to-end learning-based image compression model as illustrated in Figure 2.6. In Figure 2.6, an input image x is transformed into a latent representation $y = g_a(x, \phi)$ using the analysis transform g_a with parameter ϕ. The generated y is subjected to scalar quantization, yielding an integer vector q, corresponding to the indices of the employed scalar quantizer(s), and a quantized latent representation $\hat{y}$. At the decoder, the quantized latent representation $\hat{y}$ is transformed back into a reconstructed image in the signal domain $\hat{x} = g_s(\hat{y}, \theta)$ by means of a synthesis transform g_s with parameter θ. A chosen human visual system-based perception metric h is used to transform both the original and the decoded signal-domain images into a perceptual

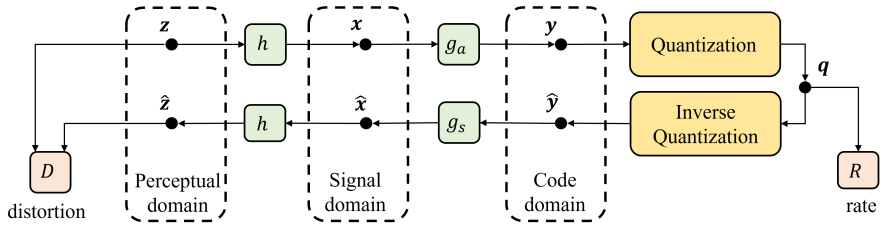

Figure 2.6: Learning-based transform coding framework proposed by Ballé *et al.* (2016b)

domain, $z = h(x)$ and $\hat{z} = h(\hat{x})$, respectively. In this framework, the analysis and synthesis transforms, g_a and g_s, are learned by optimizing a rate-distortion based cost function. In the work of Ballé *et al.* (2016a; 2016b), the generalized divisive normalization (GDN) transform which consists of a linear decomposition followed by a nonlinearity and its approximate inverse are used for g_a and g_s, respectively.

While in Ballé *et al.* (2016a; 2016b) the analysis transform takes the form of a Generalized Divisive Normalization (GDN) transform, which consists of a linear decomposition followed by a non-linear local gain control, Theis *et al.* (2017) proposed a learning-based image compression framework similar to the one of Ballé *et al.* (2016b), but where the analysis transform is implemented using a CNN architecture. Wainwright and Simoncelli (1999) showed that, for natural images, the linear filter responses such as wavelet coefficients, can be modeled as Gaussian Scale Mixtures (GSMs) (Andrews and Mallows, 1974). The GSMs were used to model the distribution of the quantized latent representation coefficients for entropy coding in the work of Theis *et al.* (2017). The GSM-based entropy model in Theis *et al.* (2017) takes the form of a fully-factorized prior whose parameters are estimated using the training image set.

For some conventional compression methods, one way to improve the entropy coding performance is to use additional bits to send side information from the encoder to the decoder. At the decoder, the side information is decoded first and used as a context while entropy decoding the image, which can help adapt the entropy coding/decoding to the local characteristics of the visual content and thus improve the coding

efficiency. Following this approach, researchers attempted to design a learning-based module that can learn information from the latent representation in order to improve the entropy coding performance. Ballé *et al.* (2018) proposed a hyperprior that is implemented by means of an autoencoder (neural network based encoder/decoder pair) that takes the latent representation as input and outputs a map with values representing local data statistics which are then used as local context information for the entropy coding.

Figure 2.7 shows the block diagram of the framework that is proposed by Ballé *et al.* (2018). As shown in Figure 2.7, the hyperprior autoencoder (represented by the encoder/decoder pair consisting of the analysis transform network h_a and the synthesis transform network h_s) outputs a spatial distribution map ($\hat{\sigma}$) representing local standard deviations, which in turn can be used as local context information by the lossless entropy coding (AE and AD in Figure 2.7, which stand for Arithmetic entropy Encoding and Arithmetic entropy Decoding, respectively). Instead of transmitting $\hat{\sigma}$ as side information, the lower bit-rate compressed output z of the hyperprior encoder h_a is quantized, losslessly compressed using entropy coding, and transmitted as side information $\hat{z}$. At the decoder, the local standard deviations $\hat{\sigma}$ can be reconstructed from the side information $\hat{z}$ using the hyperprior decoder h_s. Different from the aforementioned methods in which a fixed entropy model is used, the hyperprior component is able to adapt the entropy model based on the local characteristics of the spatial image content. This hyperprior-augmented coding method has been shown to achieve a better rate-distortion performance as compared to coding approaches that employ a non-adaptive entropy coding model.

Subsequent to the variational autoencoder with a scale hyperprior compression model of Ballé *et al.* (2018), lots of works have been done for improving the efficiency of the entropy coding model by leveraging learning-based techniques. Building on the work of Ballé *et al.* (2018), Minnen *et al.* (2018) developed a compression architecture with improved rate-distortion performance. Instead of learning the entropy context model from the hyperprior autoencoder (hyperprior encoder h_a and hyper decoder h_s), the system proposed by Minnen *et al.* (2018) learns a GSM-based probabilistic entropy coding model using an autore-

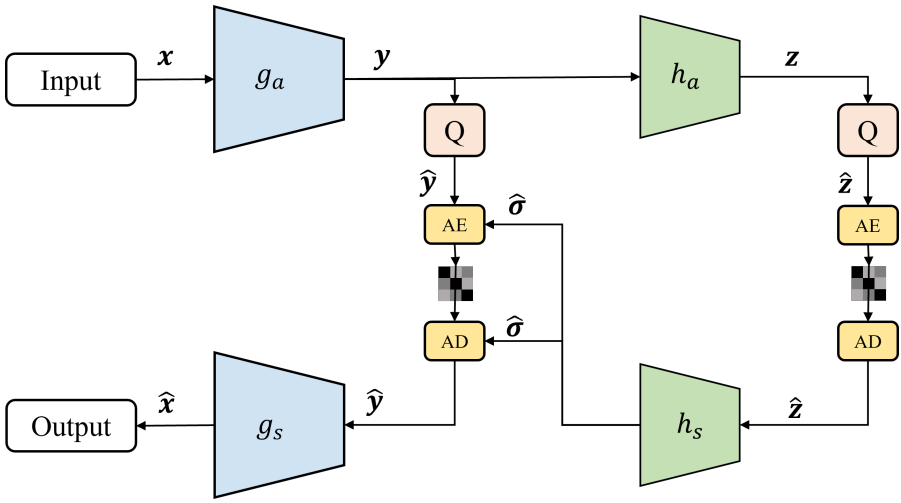

Figure 2.7: Block diagram illustrating the framework for autoencoder-based learning-based compression methods with hyperprior as described in Ballé *et al.* (2018). Q, AE, and AD stand for quantization, arithmetic entropy encoding, and arithmetic entropy decoding, respectively. For more details about the neural network architectures of the input-image autoencoder (g_a and g_s), and hyperprior autoencoder (h_a and h_s), please refer to Ballé *et al.* (2018).

gressive model over the quantized latent referred to as Context Model, along with the hyper-network (hyper encoder and hyper decoder). Considering that the Context Model can only make use of information that has already been compressed and transmitted, the Context Model is implemented using a context prediction part consisting of one single masked convolutional layer, which allows the model to use only the latent data values that have already been decoded.

The output generated by the Context Model together with the output of the hyperprior autoencoder are then used by the CNN-based Entropy Parameter network to jointly generate the mean and scale parameters of the conditional GSM-based probabilistic entropy model. All these CNN-based modules (Encoder/Decoder, Hyper Encoder/Decoder, Context Model, and Entropy Parameter network) are jointly optimized through an end-to-end training process. Lee *et al.* (2018) also proposed a compression framework based on the work of Ballé *et al.* (2018), by adapting the hyperprior autoencoder to predict the parameters of a

context-adaptive entropy model framework. In this latter framework, two types of contexts are used to estimate the standard deviation and the average parameters of a Gaussian model. Two extractors are employed to extract the two contexts from the output of the hyperprior, and from the known (previous encoded/decoded) quantized latent representation values. Instead of modeling the quantized latent into a zero-mean Gaussian in Ballé *et al.* (2018), both aforementioned networks relaxed the zero-mean restriction and model the quantized latent into a Gaussian with estimated local mean and variance.

Lin *et al.* (2020a) proposed a framework which is also based on the autoencoder with hyperprior architecture of Ballé *et al.* (2018). Instead of using one hyperprior autoencoder to estimate two parameters for the Gaussian model (Minnen *et al.*, 2018; Lee *et al.*, 2018), in order to further improve the compression performance, Lin *et al.* (2020a) modeled the local characteristics of the latent representation as a Gaussian distribution using two hyperprior autoencoder networks that estimate the local mean and variance, respectively. Building on the work of Ballé *et al.* (2018) and Minnen *et al.* (2018), Ladune *et al.* (2020) enhanced the compression performance using context-adaptive binary entropy coding employing a binary probability model instead of the commonly used Gaussian or Laplacian probability distributions.

Based on the autoencoder with hyperprior system (Figure 2.7), Liu *et al.* (2020b) proposed a learning-based image compression system called Efficient Deep Image Compression (EDIC) with a channel attention module which aims to capture the most salient features in the quantized latent representation for both the encoder and hyper encoder. As shown in Figure 2.8, the channel attention module uses a residual operation and consists of a global average pooling, a fully-connected layer and a Sigmoid layer. This channel attention module is added as part of both the encoder and hyperprior encoder after the original network. EDIC also employed a Gaussian Mixture Model (GMM), whose parameters are estimated using a network consisting of three convolutional layers and two LeakyReLU layers, referred as GMM Module, to further improve the performance of entropy modeling. Liu *et al.* (2020b) also proposed a DNN-based decoder-side enhancement module which can be used not only with EDIC, but also with other lossy image and video compression

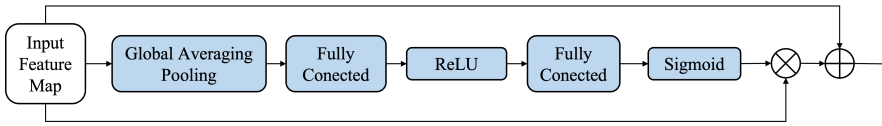

Figure 2.8: Structure for channel attention module in EDIC (Liu *et al.*, 2020b).

methods. Cheng *et al.* (2020) proposed to use discretized Gaussian Mixture Likelihoods to estimate the parameters of the entropy coding model for the quantized latent representation. A designed attention module is also used to help improve the compression performance. In both the encoder and decoder networks, two attention modules are added in the middle and at the end of each network. These attention modules can help the network to pay more attention to the important parts and to use less bits for the less important parts.

Based on their previous work (Lee *et al.*, 2018), Lee *et al.* (2019) proposed an improved network named JointIQ-Net. Lee *et al.* (2019) also proposed an improved entropy minimization method which uses a GMM for prior probability modeling. Different from the parameter estimator in Lee *et al.* (2018) which uses the reconstructed context from the hyperprior and the decoded quantized latent representation values that are adjacent to the location of the current latent representation value to be decoded, the parameter estimator in JointIQ-Net also uses a global context which is the information aggregated from both local and non-local context regions. The local context is the neighborhood within a chosen distance while the non-local context is the whole area except the local neighborhood. Additionally, JointIQ-Net jointly trains the compression model with a decoder-side post-processing enhancement network.

Based on the autoencoder-based compression model with hyperprior (Figure 2.7), Liu *et al.* (2019) proposed a learning-based image compression scheme in which a conditional context model is used in conjunction with the hyperprior decoder to predict the conditional probability distribution for a better entropy estimation as part of the entropy coding module. The authors also proposed an information compensation network (ICN) to exploit the information contained in the hyperprior latent for the final image reconstruction. This ICN takes the output of

the hyperprior decoder as input and generates a latent representation containing significant information. This generated latent representation is then fused with the entropy-decoded quantized latent representation using a concatenation operation. The resulting concatenated latent representation is input to the decoder to produce the reconstructed image.

Although the aforementioned autoencoder-based compression methods with entropy modeling (Ballé *et al.*, 2018; Minnen *et al.*, 2018; Lee *et al.*, 2018) have been shown to result in a better R-D performance than most of the popular conventional compression methods, there still exit some drawbacks such as the slow decoding speed and the large amount of computational power needed by the complex network structure. Context-adaptive models (Mentzer *et al.*, 2018; Minnen *et al.*, 2018; Lee *et al.*, 2018) may further slow down the decoding process since they are less amenable to parallelization. Several compression models have been proposed to address this problem (Minnen and Singh, 2020; Cai *et al.*, 2019). Based on the hyperprior architecture introduced in Ballé *et al.* (2018), Minnen and Singh (2020) proposed an improved system with channel conditioning and latent residual prediction. The quantized latent representation is split along the channel dimension into several slices while the whole quantized latent representation is also fed into a hyperprior autoencoder to generate the parameters of a Gaussian entropy model.

The first slice is compressed using a Gaussian entropy model conditioned only on the hyperprior while the remaining slices are compressed using a Gaussian entropy model conditioned on the hyperprior and the decoded latent representation of previous slices. Consider, for example, a quantized latent of size $W \times H \times C$. Compared with a spatially autoregressive model in which the decoding process need to be ran $W \times H$ times sequentially where only C values are computed at each iteration by splitting the latent representation into N equal-size slices along the channel, the $W \times H \times \frac{C}{N}$ values contained in each channel can be processed in parallel. Furthermore, while coding each slice, a latent residual prediction (LRP) module is used. The LRP for the first slice takes the mean parameter of the entropy model that is generated from the hyperprior as input while the LRP for the subsequent slices

takes the mean parameter as well as the decoded latent representation of previous slices as input. The outputs of the LRP module are then added to the entropy decoded latent representation to further reduce the quantization error. This method was shown to outperform other existing context-adaptive models (Minnen *et al.*, 2018; Lee *et al.*, 2018) in terms of R-D performance.

Cai *et al.* (2019) provide a learning-based sub-pixel image compression method, based on the variational autoencoder architecture with hyperprior (Ballé *et al.*, 2018), which can reduce the computational complexity. In the proposed algorithm, the input image with size $W \times H \times C$ is first transformed to a low resolution feature map with size $W/r \times H/r \times r^2C$ through a sub-pixel transform with scale parameter r, where W, H, and C correspond to the image width, height, and number of channels, respectively. Then the low resolution feature map is compressed using the compression model consisting of autoencoder with hyperprior (Ballé *et al.*, 2018). At the decoder, the reconstructed image of size $W/r \times H/r \times r^2C$ will be utilized to get the full resolution image through a proposed Sub-pixel Layer.

The aforementioned autoencoder with hyperprior architecture (Ballé *et al.*, 2018; Minnen *et al.*, 2018; Lee *et al.*, 2018) can also be used for video compression. The image compression model can be implemented on each frame separately, but this will set back the coding performance since no correlation information between frames is used. Lu *et al.* (2019a) proposed the first end-to-end Deep Video Compression (DVC) model which jointly learns motion estimation, motion compression, and residual compression using DNNs. Based on the current video compression standards, such as H.264 (Wiegand *et al.*, 2003) or H.265 (Sullivan *et al.*, 2012), DVC proposed a motion vector (MV) encoder-decoder network to estimate the optical flow and a motion compensation network to obtain the predicted frame based on the optical flow generated by the MV encoder-decoder network. The traditional linear transform / inverse transform are also replaced by a non-linear residual encoder-decoder network. A bitrate estimation net is employed in DVC for entropy coding. All components are trained jointly.

Lin *et al.* (2020b) proposed Multiple frames prediction for Learned Video Compression (M-LVC) in which not only one, but multiple previ-

ous frames are used to generate a motion vector (MV) field, resulting in a more accurate frame prediction. Based on the framework used in DVC, M-LVC introduced four new DNN-based modules: motion estimation network (ME-NET) for motion estimation and prediction, motion vector difference (MVD) encoder-decoder network to encode the difference between the original MV and predicted MV, a motion compensation network (MMC-Net) to obtain the predicted frame from the MVs of multiple previous reconstructed frames, a residual encoder-decoder network for residual compression and a residual refinement network (Residual Refine-Net) for residual refinement. Similarly, based on DVC, Lu *et al.* (2020) proposed a content-adaptive method to solve the problem of error propagation caused by the accumulation of the reconstruction error in inter predictive coding.

Lu *et al.* (2020) proposed an error propagation aware (EPA) training strategy that formulates the loss function based on a new proposed objective function that incorporates not only the RD cost of the current frame, but also the RD costs of the subsequent frames that rely on the current frame. To make the method adaptive, Lu *et al.* (2020) also proposed an online encoder updating scheme that updates the CNN parameters of the encoder based on the input while keeping the decoder unchanged. Different from the work of Lu *et al.* (2019a), the proposed method by Hu *et al.* (2021), feature-space video coding network (FVC), employs a feature extraction network to extract features from the original current frame and the reconstructed previous frames. The extracted features are later compressed, transformed, and decompressed through a video compression algorithm in the feature space. A frame reconstruction network is then employed to obtain the reconstructed frame from the decoded features. The feature-space video compression method has a structure similar to the one used for normal video compression methods. FVC replaces key components, including motion estimation, motion compression, motion compensation, and residual compression, with autoencoder style networks and all modules are jointly trained.

Park and Kim (2021) proposed an end-to-end deep predictive video compression network called DeepPVCnet. DeepPVCnet is designed to work in two different modes, uni-directional prediction in which only the two previous frames are used as reference frames to predict the current

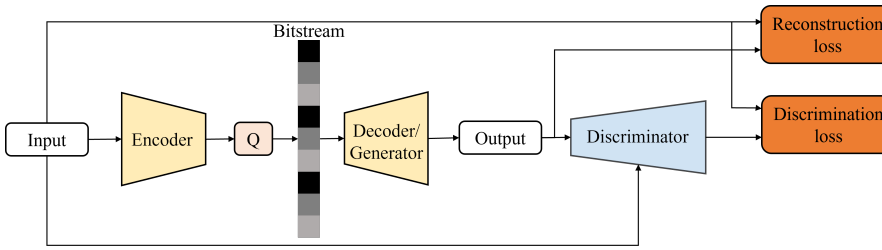

Figure 2.9: Block diagram illustrating a generative compression system.

frame, and bi-directional prediction in which two previous frames and two subsequent frames are used as reference frames to predict the current frame. Furthermore, a temporal-context-adaptive entropy coding model is proposed where context information from reference frames is used to estimate the parameters of the Gaussian entropy model.

2.2.2 Generative Compression Methods

Generative compression methods adopt a GAN-based framework into a CNN-based compression system to achieve a low compression bitrate and a high reconstruction visual quality. A generative compression system consists of an encoder, a decoder/generator and a discriminator, as shown in Figure 2.9.

Rippel and Bourdev (2017) proposed the first generative learning-based lossy image compression system. In the proposed framework, a pyramidal decomposition based encoder is employed as a feature extractor. The coefficients resulting from the decomposition are extracted at each scale. The extracted coefficient maps are then aligned using an interscale alignment procedure, to obtain a joint structure over all scales. The number of scales can be customized by the user. The extracted features (coefficients) are then quantized and entropy coded using an adaptive arithmetic codec. By considering the encoder-decoder pipeline as the generator, the authors design a discriminator that takes the target (input image) and the reconstructed image as input. The discriminator takes the form of a multiscale CNN architecture. The discriminator loss along with the reconstruction loss are used for adversarial training. Another GAN-based generative compression system was proposed in

Agustsson *et al.* (2019). It consists of an encoder, decoder/generator and a multi-scale discriminator that are jointly trained. Instead of directly using the encoder output as the latent representation, the proposed system forms the latent representation by concatenating to the encoder output a noise drawn from a fixed prior on the encoder output. The decoder/generator then tries to generate the reconstructed image from the "noisy" latent representation. Similar work was also done in Kudo *et al.* (2019). Different from the work of Rippel and Bourdev (2017) and Agustsson *et al.* (2019), in which the methods focus more on whether the reconstructed image is subjectively natural or not, the work of Kudo *et al.* (2019) proposed a generative compression system with the aim to maximize the mutual information between the coding features and the reconstructed images while still preserving good subjective naturalness. To achieve this, the authors introduce a mutual information maximizing regularization which is employed during the training.

Similar generative compression algorithms were proposed for video compression. Santurkar *et al.* (2018) proposed a neural codec architecture, referred to as NCode, which can compress both images and videos using similar system structures. In their work, the decoder network is first adversarially pre-trained using an adversarial loss with respect to the auxiliary discriminator network. Then the encoder network is trained to minimize the distortion loss. For video compression, instead of compressing the video frame-by-frame, the NCode model takes the interframe correlation into consideration. For the video compression, inspired by the interpolation scheme in MPEG4 (H.264) standard (Wiegand *et al.*, 2003) in which only every N-th frame is transmitted and the missing in-between frames are generated by means of interpolation, the author models the frames in a video sequence as uniformly-spaced samples along a path which can be approximated by linear interpolation. Thus, only the N-th frames in the manifold need to be compressed using NCode and transmitted. This procedure can be further optimized by compressing the difference between the latent of the current frame and the latent of the previous frame.

3

Survey of Datasets used for Visual Compression Methods

Most learning-based visual compression methods are accomplished through the use of deep neural networks or convolutional neural networks which need to be trained before being deployed. Most of the learning-based compression methods were trained on either the ImageNet2012 dataset or the CLIC dataset and tested on the Kodak dataset. This section introduces datasets that were used for training and testing learning-based visual compression methods. Some commonly used video datasets for end-to-end learning-based video compression methods are also introduced in this section.

3.1 Image Datasets

The datasets introduced in this section, not only include the commonly used datasets for learning-based visual compression methods, such as ImageNet2012, DIV2K, BSDS500 but also other less common datasets that were used by some of the methods described in Section 2, such as the LIVE dataset that was used in AR-CNN (Dong *et al.*, 2015) and the CSIQ dataset that was used in the GAN-based colorization network (Fatima *et al.*, 2021).

3.1.1 ImageNet2012

The ImageNet2012 dataset[1] (Russakovsky *et al.*, 2015) refers to the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)-2012 dataset which is a subset of ImageNet (Deng *et al.*, 2009). The ImageNet2012 dataset contains a total of 1,000 different classes ranging from nature creatures, like mammals, birds, insects, reptiles, amphibians, aquatic organisms, plants, etc, to inanimate artifacts, like transport, furniture, architectures, tableware, etc. The training set contains approximately 1.28 million images from the 1,000 classes; each class contains about 1,000 images. The validation dataset contains 50,000 images from the 1,000 classes; each class contains 50 images. In this dataset, one image belongs to only one specific class.

All images in the ImageNet2012 dataset are 24-RGB images stored in JPEG format. For the compression algorithms, the image used to test the compression performance are usually required to be in a lossless format. The ImageNet dataset, as constructed by its original authors (Deng *et al.*, 2009), is collected from the Internet and then each image is saved as a high-quality (high bitrate) JPG image file. So, in our tests, we consider the decoded RGB images of the ImageNet2012 dataset to be lossless.

Due to the large number of natural images covered by the ImageNet2012 dataset, learning-based compression methods trained on ImageNet2012 can result in good compression performance on most natural images. Figure 3.1 shows some example images from the ImageNet2012 validation set.

[1] ImageNet2012 dataset homepage: https://image-net.org/index.php

Figure 3.1: Example images from the ImageNet2012 validation set.

3.1.2 CLIC2020

CLIC2020[2] (Toderici *et al.*, 2020), the dataset of the CVPR workshop CLIC, is also a widely used dataset for learning-based image compression methods. The CLIC dataset comes in two versions: Dataset P ("professional") and Dataset M ("mobile"). The dataset was collected to be representative of images commonly used in the wild and includes thousands of images. Usually a mix of the professional and mobile datasets is used for training and testing. The CLIC 2020 dataset (a mixture of professional and mobile datasets) contains a total of 2,163 images in which the training set contains 1,633 imaes, the validation set contains 102 images and the testing set contains 428 images (Toderici *et al.*, 2020). The CLIC 2020 dataset contains both grayscale (1 channel, 8-bit) images and RGB (3 channels, 24-bit) images stored in PNG format.

In most cases where the CLIC dataset is used for training, the models are trained using the CLIC training and validation sets. The

[2]The CLIC2020 dataset can be used through TensorFlow: https://www.tensorflow.org/datasets/catalog/clic

Figure 3.2: Example images from the CLIC2020 dataset.

CLIC testing set is used to evaluate the performance of the trained system. Figure 3.2 shows some example images from the CLIC 2020 dataset.

3.1.3 Kodak

The Kodak[3] dataset is one of the most commonly used dataset to test the performance of image compression methods. It contains a total of 25 lossless, true color (24 bits per pixel, also known as "full color") images. All source images are stored in an uncompressed, lossless PNG format and have a size of either 768×512 or 512×768. Figure 3.3 shows example images from the Kodak dataset.

[3]The Kodak dataset homepage: http://r0k.us/graphics/kodak/

Figure 3.3: Example images from the Kodak dataset.

3.1.4 DIV2K

The DIV2K[4] (Agustsson and Timofte, 2017) is a commonly used dataset for example-based single-image super-resolution (the NITRE 2017 SR challenge). The DIV2K dataset contains 1,000 DIVerse 2K resolution high-quality images collected from the Internet. Images in DIV2K have a considerably higher resolution than many other popular image datasets. The 2K resolution means all the images in the dataset have 2K pixels on at least one of the image height and width. One common way to split the dataset is to have 800, 100, and 100 images in the training, validation and testing sets, respectively.

When used for training in learning-based compression, DIV2K is usually combined with the CLIC dataset to form a new training set. This is because the number of images in DIV2K is too small for training a deep neural network. Figure 3.4 shows example images from the DIV2K dataset.

[4]DIV2K dataset homepage: https://data.vision.ee.ethz.ch/cvl/DIV2K/

Figure 3.4: Example images from the DIV2K dataset.

3.1.5 BSDS500

Berkeley Segmentation Data Set and Benchmarks 500 (BSDS500)[5] (Arbeláez *et al.*, 2011) is a new dataset constructed as an extension of the original BSDS300 dataset. Each image in the dataset is segmented by five different subjects on average and the performance is evaluated by measuring Precision/Recall on detected boundaries and three additional region-based metrics. The dataset consists of 500 natural images, ground-truth human annotations and benchmarking code(Arbeláez *et al.*, 2011). The 500 images are separated into disjoint training set (200 image), validation set (100 images), and testing set (200 images). Figure 3.5 shows example images from the BSDS500 dataset.

[5]The BSDS500 dataset homepage: https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html

Figure 3.5: Example images from the BSDS500 dataset.

3.1.6 LIVE

The LIVE Subjective Image Quality Dataset[6] (Sheikh, 2005) was originally acquired for quality assessment (QA) research. The LIVE dataset consists of 29 resized sources images and their distorted versions. The source images in this dataset are derived from a set of source images that reflect adequate diversity in image content. There are totally 29 high resolution and high quality color images collected from the Internet and photographic CD-ROMs in this dataset (Sheikh *et al.*, 2006). The images were resized, using bicubic interpolation, to a size of 1024×768, and stored as 24bit-RGB images in bmp format. Distortions, including JPEG2000 compression, JPEG compression, white noise, Gaussian blurring and Simulated Fast Fading Rayleigh (wireless) Channel, were applied to the resized images. Figure 3.6 shows example images from the LIVE dataset.

[6]LIVE dataset homepage: https://live.ece.utexas.edu/research/quality/subjective.htm

Figure 3.6: Example images from the LIVE dataset.

3.1.7 YFCC100m

The Yahoo Flickr Creative Commons 100 Million (YFCC100m) dataset[7] (Thomee *et al.*, 2016) is the largest public multimedia collection image dataset which contains a total of 100 million media objects, and in which there are approximately 99.2 million images and 0.8 million videos. Figure 3.7 shows example images from the YFCC100m dataset.

3.1.8 Open Images

The Open Images dataset[8] (Krasin *et al.*, 2017) is a dataset used for image classification, object detection and visual relationship detection. This dataset contains approximately 9 million images that are annotated with image-level labels, object bounding boxes, object segmentation masks, visual relationships, and localized narratives. The dataset is split into a training set (9,011,219 images), a validation set (41,620 image)

[7]YFCC100m dataset Browser: http://projects.dfki.uni-kl.de/yfcc100m/

[8]Open Image dataset homepage: https://storage.googleapis.com/openimages/web/factsfigures.html

Figure 3.7: Example images from the YFCC100m dataset.

and a testing set (125,436 images). Figure 3.8 shows example images from the Open Images dataset.

3.1.9 CelebA

The CelebFaces Attributes (CelebA) dataset[9] (Liu *et al.*, 2015) is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations. The CelebA dataset contains 202,599 face images of 10,177 identities. This dataset is commonly used for visual tasks, such as fact attribute recognition, face recognition, face detection and face landmark localization. Figure 3.9 shows example images from the CelebA dataset.

3.1.10 RAISE-1k

RAw ImageS datasEt (RAISE) dataset[10] (Dang-Nguyen *et al.*, 2015) is a collection of 8,156 raw images. Images in RAISE were all captured at high resolutions (3008×2000, 4288×2848 and 4928×3264), and stored

[9]CelebA dataset homepage: http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html
[10]RAISE dataset homepage: http://loki.disi.unitn.it/RAISE/download.html#

Figure 3.8: Example images from the Open Images dataset.

in a compressed 12-bit format or a losslessly compressed 14-bit format. The images in RAISE are stored in either an sRGB (5950 images) or Adobe RGB (2206 images) color format. The RAISE-1k is a subset of the RAISE dataset which containing 1,000 images. Figure 3.10 shows example images from the RAISE-1k dataset.

3.1.11 TESTIMAGES dataset

The Tecnick TESTIMAGES dataset[11] (Asuni and Giachetti, 2014) is a large collection of sample images designed for analysis and quality assessment. The dataset contains more than 2 million images which are devided into four categories: SAMPLING, SAMPLING_PATTERNS, COLOR, and PATTERNS.

[11]TESTIMAGES dataset homepage: https://testimages.org

Figure 3.9: Example images from the CelebA dataset.

Figure 3.10: Example images from the RAISE-1k dataset.

The SAMPLING dataset mainly targets the testing resampling algorithms. This dataset contains a base of 40 RGB 2400×2400 reference im-

Figure 3.11: Example images from the TESTIMAGES-SAMPLING dataset.

ages with a bit-depth of 16 bpp and high dynamic range (HDR). As an extension of the SAMPLING dataset, the SAMPLING_PATTERNS contain 424 artificial grayscale reference images with a resolution of 1224×1224. The COLOR and the PATTERNS sets contain 8 bpp (maximum intensity 255) and 16 bpp (maximum intensity 65535) 3-channel RGB images at varying standard and non-standard resolutions. Figure 3.11 shows example images from the TESTIMAGES-SAMPLING dataset.

3.1.12 COCO

The Microsoft Common Objects in COntext (MS COCO) dataset[12] (Lin *et al.*, 2014) is a dataset used for the object detection task. This dataset contains 91 common object categories with 82 of them having more than 5,000 labeled instances. The dataset contains totally 2,500,000 labeled instances in 328,000 images. The dataset is split into a training set of 164,000 images, a validation set of 82,000 images, and a testing set of 82,000 images. Figure 3.12 shows example images from the COCO dataset.

[12]COCO dataset homepage: https://cocodataset.org/#home

Figure 3.12: Example images from the COCO dataset.

3.1.13 CSIQ

The Categorical Subjective Image Quality (CSIQ) dataset[13] (Larson and Chandler, 2010) is a dataset consisting of 30 original images and their distorted versions. Six different types of distortions, each at four to five different distortion levels, are applied to the original images. The dataset also contains 5000 subjective visual quality rating, reported

[13]CSIQ dataset homepage: https://s2.smu.edu/~eclarson/csiq.html.

Figure 3.13: Example images from the CSIQ dataset.

in the form of DMOS, from 35 observers. Figure 3.13 shows example image from the CSIQ dataset.

3.1.14 IVC

The IVC dataset[14] (Le Callet and Autrusseau, 2005) is provided by the Images and Video-communications (IVC) team at the University of Nantes. It contains 10 original images in BMP format and their distorted versions. All original images are stored in PNG format and 235 distorted images were generated from 4 different processing: JPEG, JPEG2000, LAR coding, and Blurring. This dataset also contains subjective evaluations of the distorted images. Figure 3.14 shows example images from the IVC dataset.

[14]IVC dataset homepage: https://web.archive.org/web/20200128110508/http://ivc.univ-nantes.fr/en/databases/Subjective_Database/

Figure 3.14: Example images from the IVC dataset.

3.1.15 ADE20K

The ADE20K dataset[15] (Zhou *et al.*, 2017) is a semantic segmentation dataset containing 20 thousands images annotated with 150 object categories. There are 20,210 images in the training set, 2,000 images in the validation set, and 3,000 images in the testing set. Each image in ADE20K contains at least 5 objects, and at most 273 objects. Figure 3.15 shows example images from the ADE20K dataset.

3.1.16 KITTI

The KITTI dataset[16] (Geiger *et al.*, 2013) was collected from a platform mounted on top of a car driving in and aroung Karlsruhe, Germany. The dataset contains images captured by a video camera, laser scans captured by a laserscanner, high-precision GPS measurements and IMU accelerations captured by a combined GPS/IMU system. For images, both color (24-bit RGB format) and grayscale (8-bit intensity) images

[15]ADE20K dataset homepage: https://groups.csail.mit.edu/vision/datasets/ADE20K/

[16]KITTI dataset homepage: http://www.cvlibs.net/datasets/kitti/

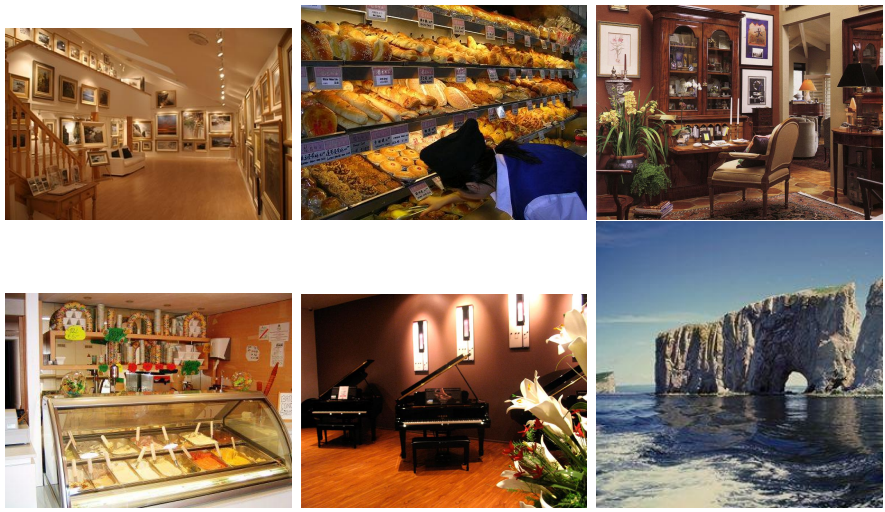

Figure 3.15: Example images from the ADE20K dataset.

Figure 3.16: Example images from the KITTI dataset.

are stored in a lossless PNG format. The original images captured by the camera have a resolution of 1392×512. Figure 3.16 shows example images from the KITTI dataset.

Table 3.1: Comparison of image datasets for learning-based visual compression methods.

Dataset	Number of Images	Resolution	Bit Depth per Channel	Color Format	Format
ImageNet	1.28 million	-	8	RGB	JPEG
CLIC2020	2,163	-	8	RGB, Grayscale	PNG
Kodak	25	768×512, 512×768	8	RGB	PNG
DIV2K	1,000	2K	8	RGB	PNG
BSDS5000	500	-	8	RGB, Grayscale	JPEG
LIVE	29, each with 5 distorted versions	1024×768	8	RGB	BMP, JPEG, JPEG2000
YFCC100m	99.2 million images, 0.8 million videos	-	-	-	-
Open Image	9 million	-	8	RGB	JPEG
CelebA	202,599	-	8	RGB	JPEG
RAISE-1k	8,156	3008×2000, 4288×2848, 4928×3264	12, 14	sRGB, Adobe RGB	TIFF
TESTIMAGES	2 million	114 standard and non-standard resolution	8, 16	RGB	PNG
COCO	328,000	-	8	RGB	JPEG
CSIQ	30, each with 5 distortion versions				
IVC	10, each with 4 distortion versions	-	8	RGB	BMP, JPEG, JPEG200
ADE20K	20 thousand	-	8	RGB	JPEG
KITTI	-	1392×512	8	RGB	PNG
UCID	1,338	-	-	-	TIFF

3.1.17 UCID

The Uncompressed Colour Image Dataset (UCID)[17] (Schaefer and Stich, 2003) is an image dataset proposed for content based image retrieval (CBIR) research. The UCID dataset contains 1,338 images stored in an uncompressed TIFF format.

Table 3.1 provides a summary of the aforementioned image datasets, including the number of images in the dataset, resolution, bit depth for each channel (e.g., color channel), color format and file format.

3.2 Video Datasets

The commonly used video datasets for training and testing learning-based video compression methods are introduced in this section.

3.2.1 Vimeo-90k

The Vimeo-90k dataset[18] (Xue *et al.*, 2019) is a large-scale, high-quality video dataset built for different video tasks such as temporal frame interpolation, video denoising, video super-resolution, and video deblocking. The Vimeo-90k contains 89,800 video clips covering large variaties of scenes and actions. All frames are resized to a fixed resolution of 448×254.

3.2.2 UVG

The Ultra Video Group (UVG) dataset[19] (Mercat *et al.*, 2020) is a dataset consisting of 16 versatile 4K test video sequences. All sequences are captured at either 50 or 120 frames per second (fps) with a resolution of 3840×2160 using a Sony F65 video camera. All video sequences are stored in a 10-bit and 8-bit color format of 4:2:0 YUV. Details of video contents, textures, descriptions and features can be found in Mercat *et al.* (2020).

[17] UCID dataset homepage: https://qualinet.github.io/databases/image/uncompressed_colour_image_database_ucid/

[18] Vimeo-90k dataset homepage: http://toflow.csail.mit.edu/

[19] UVG dataset homepage: http://ultravideo.fi/

3.2.3 VTL

The Video Trace Library (VTL) dataset[20] (VTL, 2022) contains video sequences in a 4:2:0 YUV format. All video sequences are compressed in the 7-zip format. The VTL dataset contains 20 video sequences of approximately 40,000 frames in total with a resolution of 352×288. Most of the existing works use the first 300 frames of each sequence for testing.

3.2.4 MCL-JVC

The compressed video quality assessment dataset based on the just noticeable difference model, MCL-JCV[21] (Wang *et al.*, 2016) is a widely used video quality evaluation dataset. It consists of 24 source videos with a resolution of 1920×1080 and 51 H.264/AVC encoded clips for each source sequences, with a QP ranging from 1 to 51.

3.2.5 HEVC Common Testing Sequences

The HEVC Common Testing Sequences dataset[22] (Sullivan *et al.*, 2012) was used for the development of HEVC standard and is still in use for video compression standard development. There are six classes in the HEVC Common Testing Sequences: Class A ($4K \times 2K$), Class B (1080p), Class C (WVGA), Class D (WQVGA), Class E (720p), and Class F (screen content with different resolutions). All sequences are in 4:2:0 YUV format. Most of the learning-based video compression methods use Class B, C, D, and E for testing.

3.2.6 JVET Test Sequences

The JVET Test Sequences dataset[22] (Boyce *et al.*, 2018) contains 28 test sequences that were used for the development of the reference software of the Joint Video Experts Team (JVET) for Versatile Video Coding (VVC), known as VVC Test model (VTM). The JVET test sequences

[20] VTL dataset homepage: http://trace.eas.asu.edu/yuv/index.html
[21] MCL-JCV dataset homepage: http://mcl.usc.edu/mcl-jcv-dataset/
[22] Both HEVC and JVET testing sequences are only available for qualified participants.

contain seven classes: Class A1 (4K), Class A2 (4K), Class B (1080p), Class C (WVGA), Class D (WQVGA), Class E (720p), and Class F (screen content with different resolutions). All sequences are in 4:2:0 YUV format. Most of the learning-based video compression methods use Class B, C, D, and E for testing.

3.2.7 BVI-DVC

The BVI-DVC dataset[23] (Ma *et al.*, 2021) contains 800 progressive-scanned video sequences covers different textures, natural scenes, and objects. The authors first progressive scanned 200 sequences at a resolution of 3840×2160 with frame rates between 24 fps to 120 fps from other video datasets. All sequences are stored in YCbCr 4:2:0 format with a bit depth of 10. Then these 200 sequences are spatially down-sampled to 1920×1080, 960×540, and 480×270 using Lanczos to further enlarge the dataset.

3.2.8 UGC

The UGC dataset[24] (UGC, 2022) is a large scale dataset containing YouTube User Generated Content intended for video compression and quality assessment research. The UGC dataset contains approximately 1,500 video clips with a duration of 20 seconds each. The dataset is divided into 12 categories: Animation, Cover Song, Gaming, HDR, How-To, Lecture, Live Music, Lyric Video, Music Video, News Clip, Sports, Television Clip, Vertical Video, Vlog, and VR. There are video clips with resolutions of 360P, 480P, and 1080P for all categories except HDR and VR. There are 4K video clips for HDR, Gaming, Sports, Vertical Video, Vlog, and VR genres.

3.2.9 Kinetics

The Kinetics Human Action Video dataset[25] (Kay *et al.*, 2017; Carreira and Zisserman, 2017) is a video dataset that focuses on human actions.

[23] BVI-DVC dataset homepage: https://research-information.bris.ac.uk/en/datasets/bvi-dvc

[24] UGC dataset homepage: https://media.withyoutube.com/

[25] Kinetics dataset homepage: https://www.deepmind.com/open-source/kinetics

Table 3.2: Comparison of video datasets for learning-based visual compression methods.

Dataset	Number of Sequences	Resolution	Bit Depth per Channel	Color Format
Vimeo-90k	89,800	448×254	8	-
UVG	16	4K	8	4:2:0 YUV
VTL	20	352×188		4:2:0 YUV
MCL-JVC	24	1920×1080	-	-
HEVC				
JVET	28	Class A1 (4K), Class A2 (4K), Class B (1080p), Class C (WVGA), Class D (WQVGA), Class E (720p), and Class F (screen content with different resolutions)	10	4:2:0 YUV
BVI-DVC	800	3840×2160, 1920×1080, 960×540, 480×270	10	YCbCr 4:2:0
UGC	1,500	360P, 480P, 1080P, 4K	-	-
Kinetics	306,245	-	-	-

The Kinetics dataset contains 400 human action classes, each with 400 to 1,150 video clips. Each clip was generated from a unique video and lasts around 10s. The Kinetics dataset is divided into a training set, validation set, and testing set. The testing set contains 100 video clips from each class, the validation set contains 50 video clips from each class, and the remaining video clips, about 250 to 1,000 video clips per class, are classified as part of the training set. The Kinetics dataset contains action classes including: *Person Actions (singular)* such as drinking, drawing; *Person-Person Actions* such as hugging, kissing; *Person-Object Actions* such as washing dishes, opening presents. All video clips in the Kinetics dataset are obtained from YouTube.

Table 3.2 provides a summary of the aforementioned video datasets, including number of video sequences, resolution, bit depth and color format.

4

Performance Analysis and Comparison

In this section, the performance of the aforementioned learning-based compression algorithms will be discussed. As described in Section 2, these learning-based compression algorithms can be grouped based on how the learning-based approach is applied. The performance analysis in here is done based on this grouping. Almost all the learning-based compression methods are trained and tested on one of the datasets that were presented in Section 3.

4.1 Performance Metrics

One main goal of image compression is to store and/or transmit an image using a significantly lower bitrate as compared to its original uncompressed version while maintaining a desired level of perceived visual quality. Performance measures are used for assessing the effectiveness of a compression algorithm.

4.1.1 Bitrate and Compression Ratio

To measure the achieved bitrate reduction, one popular measure is the compression ratio. The compression ratio is used to indicate the

achieved bitrate reduction and is computed by dividing the bitrate of the uncompressed image by the bitrate of the compressed image.

The per-image average bitrate (bits per pixel or bpp) is computed as:

$$bitrate = \frac{size_{bit}}{H \times W}, \tag{4.1}$$

where H and W indicate, respectively, the height and width of the image and $size_{bit}$ indicates the size of the compressed image in bits. The average bit rate over a set of images (i.e., a set of testing images or an image dataset) is calculated by averaging the bit rate over all images in the considered set.

4.1.2 Evaluation Metrics for Reconstructed Image Quality

To evaluate the perceived visual quality of a reconstructed (i.e., decompressed) image, popular visual quality assessment measures include Peak Signal-to-Noise (PSNR), Structure Similarity Index Measure (SSIM) (Wang *et al.*, 2004), and Multiscale Structure Similarity Index Measure (MS-SSIM) (Wang *et al.*, 2003). These measures are all single-channel metrics and need typically to be applied on each color channel separately. To produce a single value over all channels, the measured values over all channels can be combined (e.g., through averaging or weighted averaging).

PSNR

As a well-known quality measure for lossy compression, PSNR calculates the ratio between the maximum image value squared and the mean square error:

$$PSNR = 10 \log_{10} \frac{L^2}{MSE}, \tag{4.2}$$

in which L is the maximum value of pixel intensity ($L = 255$ for common 8-bit per pixel image format) and the mean square error MSE is calculated as:

$$MSE = \frac{1}{H \times W} \Sigma_{i=0}^{H-1} \Sigma_{j=0}^{W-1} (I_{i,j} - I'_{i,j})^2, \tag{4.3}$$

where W and H are the width and height of the image, respectively, and $I_{i,j}$ and $I'_{i,j}$ represents the intensity of the pixel at location (i, j) in the original image and the lossy decompressed image, respectively. For PSNR, a higher score means higher reconstruction quality. At very low bitrates, the PSNR has been shown not to correlate well with the perceived visual quality. Variants incorporating aspects of human visual perception such as contrast sensitivity and contrast masking include PSNR-HVS (Egiazarian *et al.*, 2006) and PSNR-HVS-M (Ponomarenko *et al.*, 2007).

For an RGB color image, the RGB-PSNR can be calculated over the whole image for the R, G and B channel separately and the final PSNR score for the image is taken as the average over the three channels.

SSIM

SSIM is an alternative and complementary approach to evaluate image quality through measuring the structural similarity between the two images. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be two discrete non-negative signals with the same size; in the image compression case, the signals are 2D matrices with pixel intensity value of non-negative integers. Let μ_x and σ_x be the mean intensity and standard deviation of $\boldsymbol{x}$, μ_y and σ_y be the mean intensity and standard deviation of $\boldsymbol{y}$, and σ_{xy} be the covariance between $\boldsymbol{x}$ and $\boldsymbol{y}$. The luminance, contrast and structure comparison measures are defined in Wang *et al.* (2004) as:

$$l(\boldsymbol{x}, \boldsymbol{y}) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}, \tag{4.4}$$

$$c(\boldsymbol{x}, \boldsymbol{y}) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}, \tag{4.5}$$

$$s(\boldsymbol{x}, \boldsymbol{y}) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}, \tag{4.6}$$

where C_1, C_2 and C_3 are small constants given by

$$C_1 = (K_1 L)^2, \quad C_2 = (K_2 L)^2, \quad C_3 = C_2/2, \tag{4.7}$$

where L is the dynamic range of the pixel values ($L = 255$ for 8-bit images) and $K_1 \ll 1$ and $K_2 \ll 1$ are two small scalar constants. K_1

and K_2 are set as 0.01 and 0.03, respectively in Wang *et al.* (2004). The SSIM between $\mathbf{x}$ and $\mathbf{y}$ is defined as:

$$SSIM(\boldsymbol{x}, \boldsymbol{y}) = [l(\boldsymbol{x}, \boldsymbol{y})]^{\alpha} \cdot [c(\boldsymbol{x}, \boldsymbol{y})]^{\beta} \cdot [s(\boldsymbol{x}, \boldsymbol{y})]^{\gamma}, \tag{4.8}$$

where $\alpha > 0$, $\beta > 0$ and $\gamma > 0$ are parameters to adjust how much the three components have an effect on the score. In Wang *et al.* (2004), the parameters are chosen as $\alpha = \beta = \gamma = 1$. With $C_3 = C_2/2$, SSIM can be expressed as:

$$SSIM(\boldsymbol{x}, \boldsymbol{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}. \tag{4.9}$$

For SSIM, a higher value indicates a higher reconstruction quality. In practice, $\boldsymbol{x}$ and $\boldsymbol{y}$ corresponds to a local region (e.g., block) in the original image and its corresponding collocated region in the compressed image, respectively. The overall SSIM score is obtained by averaging the SSIM values over all considered local image regions.

MS-SSIM

Considering that SSIM only measures the structure similarity over one scale, Wang *et al.* (2003) proposed multiscale SSIM (MS-SSIM) which is computed by calculating the SSIM over different scales. The two input signals $\boldsymbol{x}$ and $\boldsymbol{y}$ are fed into the system and a multi-scale decomposition is computed for each as described in Wang *et al.* (2003) by iteratively applying filtering operations followed by downsampling (factor of 2). This decomposition is used to get the SSIM score for different scales, resulting in MS-SSIM. Consider the original image as scale 1, the system will have M scales after $M-1$ filtering and downsampling iterations. At each scale, the contrast comparison c_j and structure comparison s_j are calculated according to Equations (4.5) and (4.6), respectively. The luminance comparison only needs to be calculated for the last scale l_M (4.4). The resulting MS-SSIM will be:

$$MS - SSIM(\boldsymbol{x}, \boldsymbol{y}) = [l_M(\boldsymbol{x}, \boldsymbol{y})]^{\alpha_M} \cdot \prod_{j=1}^{M} [c_j(\boldsymbol{x}, \boldsymbol{y})]^{\beta_j} \cdot [s_j(\boldsymbol{x}, \boldsymbol{y})]^{\gamma_j}, \tag{4.10}$$

where α_M, β_j, and γ_j are parameters used to adjust the contribution of each of the components to the final SM-SSIM value. In Wang *et al.* (2003), to simplify parameter selection, the parameters are chosen as $\alpha_j = \beta_j = \gamma_j$ and $\Sigma_{j=1}^{M}\alpha_j = 1$ and a 5-level MS-SSIM is used with weights $(0.0448, 0.2856, 0.3001, 0.2363, 0.1333)$. Similar to SSIM, a higher MS-SSIM value indicates a higher perceived visual quality of the reconstructed image.

PSNR-B

In block transform coding, the input image is divided into non-overlapping blocks of size $B \times B$ (e.g., 8×8 block as in JPEG compression) and each block is transformed and compressed independently. The block-based operation typically results in blocking artifacts occurring along the horizontal and vertical orientations. To measure the visual quality of such compressed image with blocking artifacts, Yim and Bovik (2010) proposed a new peak signal-to-noise ratio including blocking effects (PSNR-B). In PSNR-B, a blocking effect factor (BEF) is calculated to measure the blockiness of images.

Consider an image I with dimensions $N_H \times N_V$, let $\mathcal{H}$ and $\mathcal{V}$ be the set of horizontal neighboring pixel pairs and vertical neighboring pixel pairs in I, respectively. Let $\mathcal{H}_B \subset \mathcal{H}$ be the set of horizontal neighboring pixel pairs that lie across a block boundary and let $\mathcal{H}_B^C = \mathcal{H} - \mathcal{H}_B$ be the set of horizontal neighboring pixel pairs that do not lie across a boundary. Similarly, let $\mathcal{V}_B$ and $\mathcal{V}_B^C$ be the set of vertical neighboring pixel pairs that lie across a block boundary and the set of vertical neighboring pixel pairs that do not lie across a block boundary, respectively.

For a block size B, the number of pixel pairs in $\mathcal{H}_B, \mathcal{H}_B^C, \mathcal{V}_B$, and $\mathcal{V}_B^C$ are calculated as:

$$N_{H_B} = N_V \left(\frac{N_H}{B} \right) - 1, \tag{4.11}$$

$$N_{H_B^C} = N_V(N_H - 1) - N_{H_B}, \tag{4.12}$$

$$N_{V_B} = N_H \left(\frac{N_V}{B} \right) - 1, \tag{4.13}$$

$$N_{V_B^C} = N_H(N_V - 1) - N_{V_B}. \tag{4.14}$$

For a test image $\boldsymbol{y}$, the mean boundary pixel square difference (D_B) and the mean nonboundary pixel square difference (D_B^C) is defined as:

$$D_B(\boldsymbol{y}) = \frac{\Sigma_{(y_i,y_j)\in\mathcal{H}_B}(y_i - y_j)^2 + \Sigma_{(y_i,y_j)\in\mathcal{V}_B}(y_i - y_j)^2}{N_{H_B} + N_{V_B}}, \tag{4.15}$$

$$D_B^C(\boldsymbol{y}) = \frac{\Sigma_{(y_i,y_j)\in\mathcal{H}_B^C}(y_i - y_j)^2 + \Sigma_{(y_i,y_j)\in\mathcal{V}_B^C}(y_i - y_j)^2}{N_{H_B^C} + N_{V_B^C}}. \tag{4.16}$$

The blocking effect factor (BEF) is defined as:

$$\text{BEF}(\boldsymbol{y}) = \eta \cdot \left[D_B(\boldsymbol{y}) - D_B^C(\boldsymbol{y})\right] \tag{4.17}$$

where

$$\eta = \begin{cases} \frac{\log_2 B}{\log_2(\min(N_H, N_V))}, & \text{if } D_B(\boldsymbol{y}) > D_B^C(\boldsymbol{y}) \\ 0, & \text{otherwise} \end{cases}. \tag{4.18}$$

For some type of compression methods (e.g., H.264), there are multiple block sizes in the decoded image which all contribute to the blocking effect. For block size B_k, the BDR is given by:

$$\text{BEF}_k(\boldsymbol{y}) = \eta_k \cdot \left[D_{B_k}(\boldsymbol{y}) - D_{B_k}^C(\boldsymbol{y})\right]. \tag{4.19}$$

The BEF over all K block sizes is defined as:

$$\text{BEF}_{\text{Tot}}(\boldsymbol{y}) = \Sigma_{k=1}^K \text{BEF}_k(\boldsymbol{y}). \tag{4.20}$$

The MSE-B for reference image $\boldsymbol{x}$ and the test image $\boldsymbol{y}$ is defined as the sum of the $\text{MSE}(\boldsymbol{x}, \boldsymbol{y})$ and $\text{BEF}_{\text{Tot}}(\boldsymbol{y})$:

$$\text{MSE-}B(\boldsymbol{x}, \boldsymbol{y}) = \text{MSE}(\boldsymbol{x}, \boldsymbol{y}) + \text{BEF}_{\text{Tot}}(\boldsymbol{y}). \tag{4.21}$$

Then, the PSNR-B is:

$$\text{PSNR-}B(\boldsymbol{x}, \boldsymbol{y}) = 10 \log_{10} \frac{255^2}{\text{MSE-}B(\boldsymbol{x}, \boldsymbol{y})}. \tag{4.22}$$

BD-Rate

Bjøntegaard Delta-Rate[1] (Bjøntegaard, 2001) is an objective measurement used in video compression to compare rate-distortion performance

[1]The Excel Visual Basic for Applications (VBA) code for calculating RD-Rate is available at: https://github.com/tbr/bjontegaard_etro.

of two video codecs or different settings of the same video codec. BD-Rate measures the bitrate reduction offered by a codec or codec feature while maintaining the same perceived visual quality in terms of some objective measures (e.g., PSNR). The rate change is computed by averaging the rate difference in percentage over a range of quality.

4.2 Decoder-Side Post-Processing

Table 4.1 provides a summary of the experimental setup and performance results that were obtained using the decoder-side post-processing learning-based approaches described in Section 2.1.1.

The reconstructed RGB image enhancement network proposed in Lu *et al.* (2019b) is trained on the training set of DIV2K (Agustsson and Timofte, 2017) with all images compressed/decompressed by the intra coding filters of VVC and evaluated on the Test Dataset P/M with 330 images released by the Computer Vision Lab of ETC Zurich. PSNR performance on the Test Dataset with YUV 4:4:4 format achieves a 0.3 dB gain at each bitrate point and an average of 6.5% BD-Rate reduction over VVC Intra. On the Test Dataset with YUV 4:2:0 format, the network achieves a 0.5 dB gain at each bitrate point and an average of 12.2% BD-Rate reduction ober VVC Intra. The AR-CNN model proposed in Dong *et al.* (2015) is trained on a training set consisting of the training (200 images) and testing (200 images) sets of the BSDS500 dataset (Arbeláez *et al.*, 2011) and validated on the validation set of BSDS500. The LIVE dataset (Sheikh, 2005) (29 images) is used as a testing set to evaluate the quantitative and qualitative performance. Different AR-CNN models are trained for each JPEG qualities at $q = 40, 30, 20, 10$. Results show that the AR-CNN outperforms the JPEG and SR-CNN (implemented by retraining SR-CNN for JPEG) in terms of PSNR, SSIM and PSNR-B (Yim and Bovik, 2010).

For decoder-side post-processing learning-based methods for video compression, similar to Dong *et al.* (2015), the DCAD model proposed in Wang *et al.* (2017) is trained on the training (with 200 images) and testing (with 200 images) sets of the BSDS500 dataset (Arbeláez *et al.*, 2011) and validated on its validation set with 100 images. The common test condition test sequences of HEVC are used as the testing

Table 4.1: Performance analysis of decoder-side post-processing learning-based methods.

Method	Training Dataset	Validation Dataset	Testing Dataset	Anchor	Metric	Results Description
Lu *et al.* (2019b)	DIV2K (Agustsson and Timofte, 2017)	-	Test Dataset P/M (330 images) released by the Computer Vision Lab of ETC Zurich (Toderici *et al.*, 2020)	VVC Intra	PSNR	0.3 dB gains at each bitrate point and average 6.5% BD-Rate reduction for YUV 4:4:4; 0.5 dB gains at each bitrate point and average 12.2% BD-Rate reduction for YUV 4:2:0
AR-CNN (Dong *et al.*, 2015)	BSDS500 training and testing sets (Arbeláez *et al.*, 2011)	BSDS500 validation set (Arbeláez *et al.*, 2011)	LIVE dataset (Sheikh, 2005)	JPEG & SR-CNN (Dong *et al.*, 2014)	PSNR, SSIM, PSNR-B	outperforms the anchor in terms of all three metrics for JPEG compression quality settings of 10, 20, 30, 40.
DCAD (Wang *et al.*, 2017)	BSDS500 training and testing sets (Arbeláez *et al.*, 2011)	BSDS500 validation set (Arbeláez *et al.*, 2011)	Common test condition test sequences of HEVC (Bossen *et al.*, 2013)	HEVC baseline (Bossen *et al.*, 2013)	BD-rate	average 5.0%, 6.4%, 5.3%, 5.5% BD-rate reduction for coding configurations of all intra (AI), lowdelay P (LP), lowdelay B (LB), and random access (RA), respectively

Continued.

Table 4.1: Continued.

Method	Training Dataset	Validation Dataset	Testing Dataset	Anchor	Metric	Results Description
QE-CNN-I (Yang *et al.*, 2019)	BSDS500 training and testing sets (Arbeláez *et al.*, 2011)	BSDS500 validation set (Arbeláez *et al.*, 2011)	17 sequences from the JPCT-VC database (Bossen *et al.*, 2013)	AR-CNN (Dong *et al.*, 2015), VRCNN (Dai *et al.*, 2017) and DCAD (Wang *et al.*, 2017)	BD-Rate	outperforms the anchors on I frames over all testing sequences for all six QP values (22, 27, 32, 37, 42, 47)
QE-CNN-P (Yang *et al.*, 2019)	A video dataset containing 89 sequences established by the authors	-	17 sequences from the JPCT-VC database (Bossen *et al.*, 2013)	AR-CNN (Dong *et al.*, 2015), VRCNN (Dai *et al.*, 2017) and DCAD (Wang *et al.*, 2017)	BD-Rate	yields better performance on P frames over not only the anchors, but also the QE-CNN-I model
GAN-based colorization network (Fatima *et al.*, 2021)	GAN trianing set not specified by the authors	-	Kodak, IVC (Le Callet and Autrusseau, 2005), and CSIQ (Larson and Chandler, 2010) datasets	JPEG	bit rate and subjective assessment of image similarity (MOS)	outperforms JPEG at very low data rates

set. Different DCAD models are trained on four different Quantization Parameter (QP) setting (i.e., 22, 27, 32, and 37). Compared with HEVC, under four different coding configurations (all intra, low-delay, low-delay B and random access), the proposed DCAD can achieve significant improvement in terms of enhancing the reconstructed image quality and in terms of BD-rate reduction. For the QE-CNN proposed in (Yang *et al.*, 2019), QE-CNN-I for I frames is trained and validated on the same dataset as ARCNN, and QE-CNN-P for P-frames is trained on a video dataset containing 89 sequences that are provided by the authors. The two models, along with three methods are tested on 17 sequences from the JPCT-VC database (Bossen *et al.*, 2013). The proposed QE-CNN-I model outperforms the existing AR-CNN (Dong *et al.*, 2015), VRCNN (Dai *et al.*, 2017) and DCAD (Wang *et al.*, 2017) methods on I frames over all testing sequences for all six QP values chosen by the authors (22, 27, 32, 37, 42, 47). The proposed QE-CNN-P model yields better performance on P frames as compared not only to the aforementioned three methods, but also to the QE-CNN-I model.

The GAN-based colorization network proposed in Fatima *et al.* (2021) is evaluated on the Kodak, IVC, and Computational and Subjective Image Quality (CSIQ) datasets in terms of bit rate and image quality. Results show that the GAN-based colorization network can outperform JPEG at very low data rates.

Overall, for a given bit-rate, the decoder-side post-processing approaches can achieve a higher visual quality for the reconstructed images. Such improvement can also be seen on video compression methods with decoder-side post processing.

The advantage of decoder-side post-processing approaches is that the learning-base enhancement module can be added in conjunction to any conventional compression method. With proper training, the post-processing module can help in improving the reconstruction quality.

One problem with decoder-side post-processing approaches is that the performance of the learning-based post-processing module relies on how good the trained model is. Thus, different models need to be trained for different conventional compression methods, even for different quality levels of the same conventional compression methods.

4.3 Encoder-Side Pre-Processing

Table 4.2 provides a summary of the experimental setup and performance results that were obtained using the encoder-side pre-processing learning-based approaches described in Section 2.1.2.

As discussed before, most encoder-side pre-processing approaches are paired with decoder-side post-processing. The most popular pre- and post-processing pair is the color space conversion. The ABC model (Li, 2019) implemented on H.266 (VTM 4.0) is tested on the CLIC2019 validation dataset at high QP 34 $\sim$ 38 and low QP 20 $\sim$ 24. Results show that the proposed method, with the PCA on encoder-side and LSM on decoder-side, gives significant improvement in terms of an RGB-PSNR boost of 0.26 dB gain at 0.145 bpp as compared to the baseline, which is obtained by using the traditional BT.601 color space conversion and H.266 (VTM 4.0) with all tools enabled on input images (YCbCr 4:2:0 format). It also results in a 1.2 dB gain at 1.0 bpp as compared to H.266 (VTM 4.0). The later proposed VimicroABCnet (Li *et al.*, 2019) is also tested on the CLIC2019 validation set. Similar to ABC net in Li (2019), the VimicroABC net is compared with H.266 with BT.601 conversion. Results show that the proposed enhancing network in VimicroABC net can achieve a boost of 0.3 dB gain in RGB-PSNR at 10.15 bpp. Combined with ABC net, the VimicroABC net can achieve even better performance achieving a RGB-PSNR gain of 0.56 dB gain at 0.15 bpp as compared to H.266. Overall, VimicroABCnet can achieve a smaller compressed file with a higher reconstructed quality in terms of RGB-PSNR.

The VIP-ICT codec (Sun *et al.*, 2020) is built based on VimicroABC-net and is trained on the CLIC 2020 training set and DIV2K (Agustsson and Timofte, 2017) dataset with a patch size of 64×64. It is tested on the CLIC 2020 testing set. Compared with the anchor generated by VTM 7.1 without built-in filters, the trained model gives about 1.0 to 1.5 dB improvement as compared to the anchor in terms of PSNR at all the considered bitrates.

The three aforementioned learning-based color format conversion pre-/post-processing pair can all get improved reconstruction quality in terms of PSNR as compared to the basic conventional compression

Table 4.2: Performance analysis of encoder-side pre-processing learning-based methods.

Method	Training Dataset	Testing Dataset	Anchor	Metric	Results Description
ABC (Li, 2019)	-	CLIC 2019 validation dataset (Toderici *et al.*, 2020)	H.266 (VTM 4.0) with BT.601 conversion (ITU, 2011)	RGB-PSNR	RGB-PSNR boost of 0.26 dB @0.145 bpp, and 1.2 dB @1.0 bpp
Vimicro-ABCnet (Li *et al.*, 2019)	CLIC2018 train and test sets (Toderici *et al.*, 2020)	CLIC 2019 validation dataset (Toderici *et al.*, 2020)	H.266 with BT.601 conversion	RGB-PSNR	RGB-PSNR boost of 0.56 dB @0.15 bpp
VIP-ICT codec (Sun *et al.*, 2020)	CLIC 2020 training set and DIV2K (Agustsson and Timofte, 2017) dataset	CLIC 2020 testing set	H.266 (VTM 7.1) without built-in filters	PSNR	gives about 1.0 to 1.5 dB improvement for different bitrates over the anchor
CrCNN and ReCNN pair (Tao *et al.*, 2017)	BSDS500 dataset (Arbeláez *et al.*, 2011)	Standard test image “Butterfly”, “Cameraman”, “House”, “Lena”, “Peppers”, “Leaves” and “Parrots”	JPEG and JPEG2000	PSNR, SSIM	achieves better compression performance over the anchors in terms of visual quality
DSSLIC (Akbari *et al.*, 2019)	ADE20K (Zhou *et al.*, 2017) training set	ADE20K (Zhou *et al.*, 2017) test set and the Kodak set	JPEG, JPEG2000, WebP, and BPG	MS-SSIM in RGB domain	achieves higher visual quality

methods they are built on. One problem with the reported results is that all results are reported in terms of PSNR while recent research shows that compared with SSIM and MS-SSIM, PSNR is less correlated with the human visual system, especially at low bitrate.

Similar to the super-resolution methods that are used in decoder-side post-processing, the de-resolution (CrCNN) and super-resolution (ReCNN) pair in Tao *et al.* (2017) showed better compression performance in terms of both PSNR and SSIM over JPEG and JPEG2000 on standard test images including "Butterfly", "Cameraman", "House", "Lena", "Peppers", "Leaves" and "Parrots". The DSSLIC (Akbari *et al.*, 2019) framework, due to its property in using segmentation to improve compression performance, uses a segmentation dataset, ADE20K (Zhou *et al.*, 2017) with 150 semantic labels, for training. The trained model is then tested on the ADE20K test set (averaged over 50 random test images not included in training) and the Kodak set. Compared with JPEG, JPEG2000, WebP, and BPG, the proposed framework can achieve a higher visual quality in terms of PSNR and MS-SSIM in the RGB domain.

Compared with color space conversion methods, the de-resolution/super-resolution pair based pre-processing/post-processing pair can give not only improvement in PSNR, but also in SSIM and MS-SSIM.

The encoder-side pre-processing methods have similar advantages as the post-processing ones since they can be applied on different conventional compression methods with proper training. They also share a similar disadvantage is that different models need to be trained for different compression methods.

4.4 DNN-based Modules as part of Conventional Codecs

Table 4.3 provides a summary of the experimental setup and performance results that were obtained using the aforementioned learning-based approaches in which DNN-based modules are used to replace parts of a conventional codec as described in Section 2.1.3.

The VRCNN (Dai *et al.*, 2017), built based on the 4-layer artifact reduction network AR-CNN (Dong *et al.*, 2015), is used to replace the in-

Table 4.3: Performance analysis of DNN-based modules as part of conventional codecs.

Method	Training Dataset	Testing Dataset	Anchor	Metric	Results Description
VRCNN (Dai *et al.*, 2017)	BSDS500 training and testing sets (Arbeláez *et al.*, 2011)	HEVC standard test sequences	HEVC, AR-CNN (Arbeláez *et al.*, 2011)	BD-rate	achieves an average 4.6% DB-rate reduction on the test sequences as compared to the HEVC baseline
IFCNN (Park and Kim, 2016)	8 sequences from the HEVC standard test sequences	8 sequences from the HEVC standard test sequences	HEVC	BD-rate, PSNR	yields average 4.8% BD rate reduction and achieves better performance in terms of PSNR
CNN-based image compression scheme (Ma *et al.*, 2019b)	DIV2K compressed with JPEG2000	Kodak dataset	JPEG2000	BD-rate	achieves average BD-rate reduction of 23.68% over bitrates 0.25 bpp, 0.5 bpp, 0.75 bpp, and 1.0 bpp. Reduction can be as high as 36.34% on image Kodak05
Angular intra-prediction methods in HEVC (Schiopu *et al.*, 2019)	ADE20K	HEVC test sequences (Class A, B, C, and D), UVG dataset	Lossless HEVC Intra, AP-CNN (Huang *et al.*, 2019)	BD-rate	achieve a bitrate reduction of 4.798% on average for the HEVC test sequences dataset and 5.822% on average for the UVG dataset
GAN-based Intra-prediction (Zhu *et al.*, 2019)	UCID (Schaefer and Stich, 2003)	HEVC test sequences	HEVC test model (HM 16.17), Li *et al.* (2018a)	BD-rate	outperforms the HEVC baseline and work of Li *et al.* (2018a) when implementing based on HEVC, outperforms VVC baseline when implementing based on VVC
Benjak *et al.* (2021)	a dataset collected by the authors	KITTI (Geiger *et al.*, 2013)	VTM 7.0	BD-rate	achieves an average BD-rate reduction of 0.94% over the VTM 7.0 baseline. But due to the computational complexity of neural network, the encoding time increases by 5% while the decoding time increases by 300%

Continued.

Table 4.3: Continued.

Method	Training Dataset	Testing Dataset	Anchor	Metric	Results Description
FRCNN (Yan *et al.*, 2018)	*BlowingB-ubbles* (one test sequence from the HEVC dataset, Sullivan *et al.*, 2012)	HEVC dataset (Sullivan *et al.*, 2012)	HEVC (HM 16.7)	BD-rate	achieves 3.9%, 2.7%, and 1.3% for luma component on Lowdelay-P, Lowdelay-B, and Random Access modes in terms of BD-rate
Wu *et al.* (2018)	Kinetics dataset (Carreira and Zisserman, 2017)	Kinetics (Carreira and Zisserman, 2017), VTL (VTL, 2022), and UVG (Mercat *et al.*, 2020) datasets	HEVC, H.264, MPEG-4 Part 2, and H.261	bit per pixel (bpp), MS-SSIM, and PSNR	outperforms H.261 and MPEG-4 Part 2, achieves camparable results to H.264 and HEVC. For high resolution videos in UVG dataset, the proposed model outperforms H.264 and gives comparable performance with HEVC in terms of PSNR
CNNAC for DC coefficients Ma *et al.* (2018)	UCID (Schaefer and Stich, 2003) and DIV2K (Agustsson and Timofte, 2017) datasets	HEVC dataset (Sullivan *et al.*, 2012)	HEVC (HM 12.0)	bitrate saving, BD-rate	achieves an average bit-rate saving of 22.47% for DC coefficients. In terms of BD-rate, the proposed model can achieve 1.8% reduction on average
CNNAC for HEVC intra-predicted residuals (Ma *et al.*, 2019a)	UCID (Schaefer and Stich, 2003) and DIV2K (Agustsson and Timofte, 2017) datasets	HEVC dataset (Sullivan *et al.*, 2012)	HEVC (HM 12.0)	bitrate saving, BD-rate	achieves BD-rate reduction for Y, U, and V channels of 4.7%, 4.2%, and 4.0% on average, respectively.

loop deblocking filter and SAO in HEVC. The VRCNN model is trained on a collection of 400 natural images (BSDS500 training and testing sets (Arbeláez *et al.*, 2011) each contains 200 images), which are the same images as in Dong *et al.* (2015), and is tested on the HEVC standard test sequences, which contains 20 sequences of 5 classes. Four separate networks are trained on images compressed by HEVC intra coding (without deblocking and SAO) with different QP values (i.e., 22, 27, 32, and 37). VRCNN can achieve an average of 4.6% DB-rate reduction on the test sequences, which outperforms both HEVC and AR-CNN. But since the VRCNN architecture is more complex than AR-CNN, its decoding procedure takes more time. IFCNN proposed by Park and Kim (2016) replaces only SAO in HEVC, and uses 8 sequences from the HEVC standard test sequences for training and testing. Results show that, compared with the original HEVC, the proposed IFCNN yields an average 4.8% BD rate reduction and achieves better performance in terms of PSNR.

In Ma *et al.* (2019b), the DIV2K (Agustsson and Timofte, 2017) dataset compressed with the JPEG2000 (Jpaser, Adams, 2006) was used for training while the Kodak dataset was used for testing. Experiments are performed to compare the proposed method with JPEG2000 at bitrates of 0.25 bpp, 0.5 bpp, 0.75 bpp, and 1.0 bpp. The average BD-rate reduction is 23.68%, and it can be as high as 36.34% on the image Kodak05.

Schiopu *et al.* (2019) trained the modified AP-CNN model for each of the 33 angular intra-prediction modes in HEVC. The training set used for each model is the corresponding set containing input patches generated based on HEVC's optimal mode segmentation of frames extracted from the ADE20K dataset. Using the Losselss HEVC Intra as baseline, the proposed method can achieve a bitrate reduction of 4.798% on average for the HEVC test sequences dataset and of 5.822% on average for the UVG dataset. Schiopu *et al.* (2019) also proposed a hybrid method in which the optimal block segmentation is generated based on the competition between the HEVC-based prediction and the CNN-based prediction. By combining the HEVC-based prediction and the CNN-based prediction, the performance can be further improved to result in a bitrate reduction of 5.146% for the HEVC dataset and of

5.998% for the UVG dataset. Zhu *et al.* (2019) trained the proposed GAN model on the UCID dataset (Schaefer and Stich, 2003). The proposed method applied to the HEVC test model (HM 16.17) outperforms the HEVC baseline as well as the method of Li *et al.* (2018a) for all the Y, U, V channels on average. Compared with the HEVC baseline, the GAN-based intra prediction model can achieve a bitrate reduction of 6.6%, and 7.5% on average for the luma and two chroma components, respectively. The authors also integrated the proposed method with the VVC Test Model (VTM 1.1). Experimental results show a bitrate reduction of 3.10%, 6.75%, and 6.83% for the luma component for small, normal, and large QP values, respectively, as compared to the VVC baseline.

The enhanced learning-based inter coding algorithm proposed by Benjak *et al.* (2021) was trained on a dataset that contains 487,020 pictures, which are generated by downsampling 13 YouTube HD sequences of train and subway drives filmed with a front-viewing camera. The trained model was then evaluated on the KITTI dataset (Geiger *et al.*, 2013). The proposed model when integrated with VVC can achieve an average BD-rate reduction of 0.94% over the VTM 7.0 baseline. But due to the computational complexity of the neural network, the encoding time increases by 5% while the decoding time increases by 300%. FRCNN (Yan *et al.*, 2018) was trained on a training dataset generated by the authors from *BlowingBubbles* (one test sequence from the HEVC testing sequences dataset, Sullivan *et al.*, 2012) using the HEVC encoder-HM 16.7. Compared with the HEVC anchor, the proposed FRCNN can achieve significant bitrate savings in terms of BD-rate reductions of 3.9%, 2.7%, and 1.3% for the luma component using the Lowdelay-P (LDP), Lowdelay-B (LDB), and Random Access (RA) modes, respectively. FRCNN also resulted in a BD-rate reduction of around 1% for the chroma components for the LDP, LDB, and RA modes.

Wu *et al.* (2018) trained the proposed model on selected videos with width and height larger than 720px from the Kinetics dataset (Carreira and Zisserman, 2017) and tested on the Kinetics (Carreira and Zisserman, 2017), VTL (VTL, 2022), and UVG (Mercat *et al.*, 2020) datasets. Wu *et al.* (2018) compared their model with HEVC, H.264,

MPEG-4 Part 2, and H.261. The proposed model outperforms H.261 and MPEG-4 Part 2, and achieves comparable results with H.264 and HEVC. For high resolution videos in the UVG dataset, the proposed model outperforms H.264 and gives a comparable performance with HEVC in terms of PSNR.

Ma *et al.* (2018) trained their proposed CNNAC model on the UCID (Schaefer and Stich, 2003) and DIV2K (Agustsson and Timofte, 2017) datasets. The training dataset is prepared using the HEVC HM 12.0 codec with all-intra mode. Tested on the HEVC testing dataset, using the HEVC HM 12.0 intra codec as anchor, the proposed CNNAC for DC coefficients can achieve an average bitrate saving of 22.47% for DC coefficients. In terms of BD-rate, the proposed model can achieve a 1.8% reduction on average. The extended CNNAC model for HEVC intra-predicted residuals (Ma *et al.*, 2019a) was also trained on the UCID (Schaefer and Stich, 2003) and DIV2K (Agustsson and Timofte, 2017) datasets. The proposed model achieves a BD-rate reduction for the Y, U, and V channels of 4.7%, 4.2%, and 4.0% on average, respectively, with HM 12.0 used as anchor.

These results show that through using learning-based modules in lieu of parts of the conventional compression methods, the resulting compression methods can achieve a better compression performance in terms of BD-rate. Different from the two previous pre- and post-processing learning-based compression methods, the DNN-based module used in lieu of parts of a conventional codec, is less transportable.

4.5 Autoencoder-based Approaches

4.5.1 Autoencoder-based Codecs with Fixed Entropy Model

Table 4.4 provides a summary of the experimental setup and performance results that were obtained using the autoencoder-based approaches with the basic autoencoder structure as described in Section 2.2.1.

Toderici *et al.* (2015) used a dataset containing 216 million color images that were collected from the public internet, downsampled to a size of 32×32, and stored in a lossless format, for training and testing. A subset of 100k randomly selected images from this dataset is

Table 4.4: Performance analysis of autoencoder-based codecs with fixed entropy model.

Method	Training Dataset	Testing Dataset	Anchor	Metric	Results Description
Toderici *et al.* (2015)	dataset contains 216 million images collected from the public internet not including the testing subset	a subset of 100k randomly selected images from this dataset	JPEG, WebP	bitrate, SSIM	achieves equal or better perceived visual quality level with 4% to 12% lower average bitrate than the anchors
Toderici *et al.* (2017)	dataset from Toderici *et al.* (2015), and "High Entropy" (HE) dataset containing 6 million 1280×720 images from the web	Kodak	JPEG (4:2:0)	PSNR-HVS, MS-SSIM	outperforms the anchor across most bitrates on the R-D curve on the testing dataset, and achieves better performance in terms of both MS-SSIM AUC (area under the R-D curve) and PSNR-HVS AUC
Johnston *et al.* (2018)	same 6 million 1280×720 dataset as Toderici *et al.* (2017)	Kodak and Tecnick SAMPLING dataset	BPG (4:2:0), JPEG2000, WebP, JPEG, Theis *et al.* (2017), and Toderici *et al.* (2015)	MS-SSIM	outperforms the anchors in terms of MS-SSIM at different bitrates
Ballé *et al.* (2016a)	ImageNet	Kodak	JPEG, JPEG2000	bitrate, luma MS-SSIM and luma PSNR	outperforms the anchors over a varying range of compression bitrates
Theis *et al.* (2017)	434 high quality images from Flickr website (Flickr, 2022)	Kodak	JPEG, JPEG2000, Toderici *et al.* (2017)	PSNR, SSIM, MS-SSIM	outperforms all anchors in terms of SSIM, achieves similar results compared to all anchors in terms of MS-SSIM, performs similar to JPEG2000 in terms of PSNR

used for evaluating the proposed codecs. The (de)convolution LSTM model exhibits perceptual quality levels that are equal to or better than both JPEG and WebP at a 4% to 12% lower bitrate on average. The authors later improved their work and proposed an RNN-based encoder/decoder with binarizer in Toderici *et al.* (2017). This improved framework uses the same "32 × 32" dataset as their previous work (Toderici *et al.*, 2015) for training and also makes use of a second dataset containing 6 million 1280 × 720 images from the web, referred to as "High Entropy" (HE) dataset for training. The second dataset is decomposed into non-overlapping 32 × 32 tiles before being used. Separate models are trained on each dataset and all trained models are tested on the Kodak Photo CD dataset and compared with JPEG (4:2:0). The obtained experimental results show that, in terms of both PSNR-HVS and MS-SSIM, models trained on both datasets have a better RD-curve as compared to JPEG (4:2:0). Since the second "High Entropy (HE)" training dataset is designed to be "hard-to-compress", results show that models trained on the "32 × 32" dataset outperforms the models trained on the "HE" dataset in terms of PSNR-HVS and MS-SSIM.

The learning-based compression methods proposed in Johnston *et al.* (2018) also used the 6 million 1280×720 "HE" dataset but with 128×128 patches randomly sampled from the images. The trained model was then tested on the Kodak dataset and the Tecnick (Asuni and Giachetti, 2014) SAMPLING dataset and compared with BPG (4:2:0), JPEG2000, WebP, JPEG and the work of Theis *et al.* (2017) and Toderici *et al.* (2015). The Tecnick SAMPLING dataset which contains 100 200 × 200 images was used since its results can be more representative of contemporary, high resolution content. Compared with the aforementioned conventional compression methods, the proposed framework gave better performance in terms of MS-SSIM for the tested bitrate levels. Compared with the two autoencoder-based methods from Theis *et al.* (2017) and Toderici *et al.* (2015), the proposed framework outperforms these for all bitrates. By further adding the proposed spatially adaptive bit rate (SABR) post-processing, the models of Johnston *et al.* (2018) can achieve better compression performance even when the models are not retrained to handle SABR. This shows that a better performance can be expected if the

models are further improved by performing some retraining for SABR.

Although these methods can achieve better performance than most of the conventional compression methods, the methods focusing on how to generate high-quality reconstructed images from a given fixed number of representations did not lead to further improvement in later years. More recent methods were able to achieve an increase in the R-D performance by learning a pair of analysis and synthesis transforms while performing a rate-distortion optimization.

The proposed end-to-end image compression network based on nonlinear transform coding in Ballé *et al.* (2016a) was trained on the ImageNet dataset (Deng *et al.*, 2009) and tested on the Kodak dataset. Evaluated with respect to bitrate and reconstruction quality in terms of luma MS-SSIM and luma PSNR, the trained model outperformed both JPEG and JPEG2000 for a range of compression bitrates. A similar framework was proposed in Theis *et al.* (2017) and was trained on a dataset consisting of 434 high quality images from the Flickr website (Flickr, 2022). It was then tested on the Kodak dataset. The proposed autoencoder outperformed not only the conventional JPEG compression method, but also the end-to-end learning-based compression method of Toderici *et al.* (2017) in terms of PSNR and SSIM. In terms of MS-SSIM, the proposed autoencoder produced a very similar score as the other compared methods (i.e., JPEG, JPEG2000, and method of Toderici *et al.*, 2017).

The aforementioned end-to-end learning-based autoencoder methods with rate-distortion performance optimization yield improvement as compared to conventional compression methods such as JPEG and JPEG2000. But such methods are limited by the fact that the parameters of the entropy model is fixed for different inputs. Intuitively, an adaptive entropy model with different parameters for different inputs could achieve better compression performance.

4.5.2 Autoencoder-based Codecs with Hyperprior

Table 4.5 gives a summary of the experimental setup and the obtained performance results for the autoencoder-based approaches with hyperprior as described in Section 2.2.1.

Table 4.5: Performance analysis of autoencoder-based codecs with hyperprior.

Method	Training Dataset	Testing Dataset	Anchor	Metric	Results Description
Ballé *et al.* (2018)	a dataset containing approximately 1 million color JPEG images with height/width between 3,000 and 5,000 pixels randomly scraped from the internet	Kodak	JPEG (4:2:0), BPG (4:4:4), methods from Rippel and Bourdev (2017), Ballé *et al.* (2016a), and Theis *et al.* (2017)	rate-distortion in terms of PSNR, MS-SSIM	in terms of PSNR, the models optimized for MSE give comparable results as BPG and outperform all the other anchors. In terms of MS-SSIM, the models optimized for MS-SSIM give the best results as compared to all the anchors
Minnen *et al.* (2018)	-	Kodak	JPEG, JPEG 2000 and BPG (4:4:4), and Ballé *et al.* (2018)	rate-distortion in terms of PSNR, MS-SSIM	in terms of PSNR, the models optimized for MSE outperform all the anchors and in terms of MS-SSIM, the models optimized for MS-SSIM outperforms all the anchors
Lee *et al.* (2018)	256×256 patches extracted from 32,410 images randomly selected from YFCC100m (Thomee *et al.*, 2016)	Kodak	BPG and JPEG 2000, and Ballé *et al.* (2018) and Theis *et al.* (2017)	rate-distortion in terms of PSNR, MS-SSIM	in terms of PSNR, the models optimized for MSE yield a comparable performance as BPG and outperform all the other anchors, and in terms of MS-SSIM, the models optimized for MS-SSIM outperforms all the anchors
Lin *et al.* (2020a)	a set of 20 thousand images which are chosen from the DIV2K and ILSVRC2012	Kodak	BPG, Minnen *et al.* (2018), Ballé *et al.* (2018), Lee *et al.* (2018), and Choi *et al.* (2019)	rate-distortion in terms of PSNR, MS-SSIM	outperforms all the anchors in terms of both PSNR and MS-SSIM
Ladune *et al.* (2020)	DIV2K and CLIC datasets	CLIC 2019 validation and test sets	BPG	bitrate	the proposed binary probability model yields 9.1% rate saving while achieving competitive results with BPG

The autoencoder-based codec with hyperprior framework in Ballé *et al.* (2018) was trained on a dataset which contains approximately 1 million color JPEG images with height/width between 3,000 and 5,000 pixels randomly sampled from the Internet. Similar to other autoencoder-based methods, the trained models were tested on the Kodak dataset. The authors trained 32 different models, with/without hyperprior, optimized for MSE or MS-SSIM as distortion metric. For each combination, 8 models were trained for 8 different bitrate levels (i.e., one model for each desired bitrate level). The obtained results showed that models with hyperprior outperform models without hyperprior for both optimization metrics over all bitrate levels. The trained models were compared with JPEG (4:2:0), BPG (4:4:4), the methods from Rippel and Bourdev (2017), Ballé *et al.* (2016a) and Theis *et al.* (2017).

Models with hyperprior optimized for MSE gave the best performance when PSNR was used to evaluate the reconstructed image quality as compared to most of the selected existing codecs and gave comparable results to BPG. Models with hyperprior optimized for MS-SSIM gave the best performance when MS-SSIM was used to evaluate the reconstructed image quality as compared to the selected existing codecs. The variational autoencoder with hyperprior compression framework became one of the most popular baseline framework for developing end-to-end learning-based compression approaches. Many improvements were proposed based on this architecture. Figure 4.1 shows the visual examples of compressed images that were obtained using the model of Ballé *et al.* (2018) with the model optimized using MS-SSIM. For comparison, Figure 4.1 also shows compressed images that were obtained using conventional compression methods including JPEG2000 (Kakadu), JPEG (libjpeg), and HEVC Intra (HM 16.25) for different bitrates.

The compression model proposed in Minnen *et al.* (2018) was trained and then evaluated on the Kodak dataset in terms of rate-distortion performance using PSNR/MS-SSIM as the reconstruction quality metric. In terms of PSNR, the proposed method which combines a Context Model and hyperprior (optimized for MSE) outperformed not only the conventional compression methods, such as JPEG, JPEG 2000 and BPG (4:4:4), but also the learning-based compression method

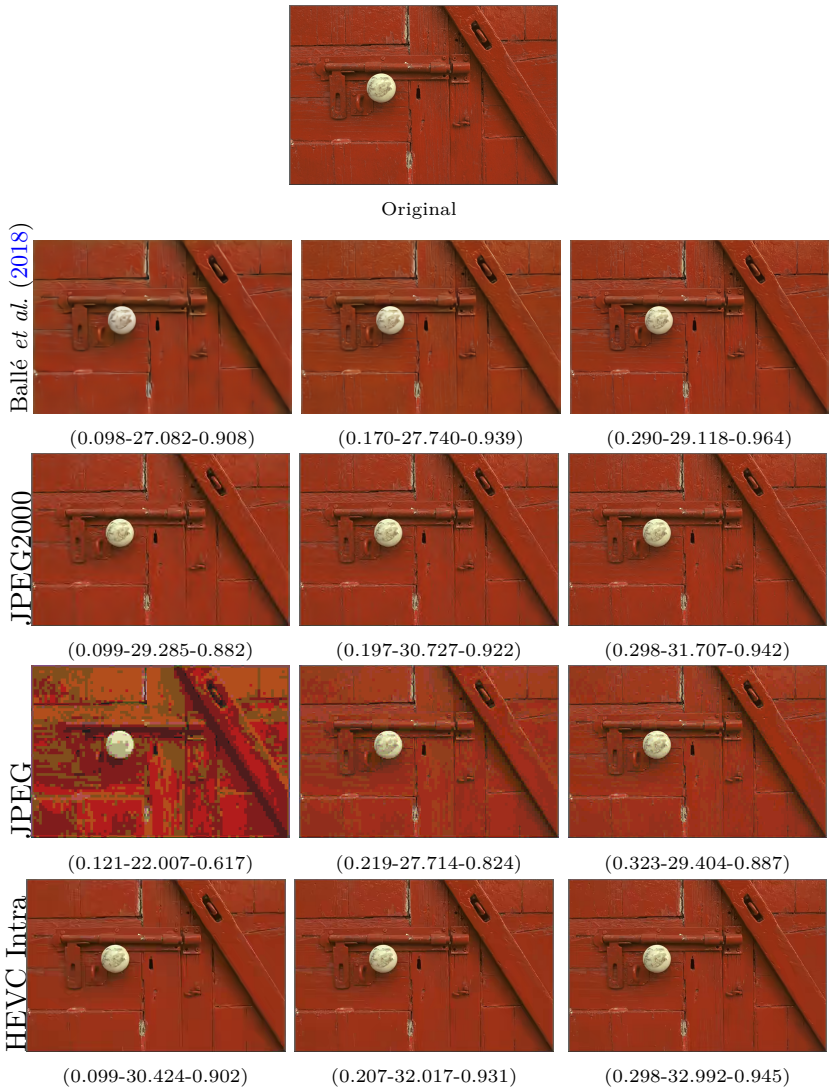

Figure 4.1: Visual compression results for the 768×512 "kodim02" image from the Kodak dataset: Original image (row 1), compressed using Ballé *et al.* (2018) (row 2), JPEG2000 (row 3), JPEG (row 4), HEVC Intra (implemented using HM 16.25) (row 5) for different target bit rates of 0.1 bpp (left column), 0.2 bpp (middle column), and 0.3 bpp (right column). The performance measures are shown under each image as (bitrate in bpp - PSNR in dB - MS-SSIM).

in Ballé *et al.* (2018). When optimized for MS-SSIM, the proposed method gave the best performance in terms of MS-SSIM as compared to the MS-SSIM optimized model of Ballé *et al.* (2018) and Rippel and Bourdev (2017). Another improved framework was proposed by Lee *et al.* (2018), and which was based on the architecture of Ballé *et al.* (2018). This latter framework used a dataset containing 256×256 patches extracted from 32,410 images that were randomly selected from the YFCC100m (Thomee *et al.*, 2016) dataset for training and the Kodak dataset for evaluation. Similar to Ballé *et al.* (2018), different models with hyperpriors optimized for MSE and MS-SSIM, were trained and compared with both conventional compression methods (BPG and JPEG 2000), and learning-based compression methods (work of Ballé *et al.*, 2018 and Theis *et al.*, 2017). The obtained performance results were similar to the ones obtained using the method of Ballé *et al.* (2018) in terms of PSNR, with the model optimized for MSE yielding an improved performance as compared to the existing codecs except for BPG which yielded a comparable performance. In terms of MS-SSIM, the proposed model optimized for MS-SSIM gave the best performance as compared to all the mentioned existing codecs.

Lin *et al.* (2020a) used a set of 20,000 images that were chosen from the DIV2K (Agustsson and Timofte, 2017) and ILSVRC2012 datasets for training their proposed BRNN model. The Kodak dataset was used for testing. The proposed BRNN model outperformed BPG and other learning-based methods, including the methods of Minnen *et al.* (2018), Ballé *et al.* (2018), Lee *et al.* (2018) and Choi *et al.* (2019), in terms of both PSNR and MS-SSIM. When using PSNR as the visual quality metric, the improvement was not very significant while, when using MS-SSIM as the visual quality metric, the learning-based compression methods resulted in a significantly higher MS-SSIM score than BPG. The compression model proposed in Ladune *et al.* (2020) was trained on the DIV2K and CLIC datasets and tested on the CLIC 2019 validation and test sets, which contain 102 and 330 images of various resolutions, respectively. The proposed method incorporated a binary probability model for the latent variables, and it resulted in significant bitrate savings of up to 18.3% as compared to a Gaussian probability model for a lightweight system (in which the inside convolutional layers have

64 kernels). Experimental results also showed that the proposed binary probability model yields up to 9.1% in bitrate savings as compared to the Gaussian probability model for a standard system (in which the inside convolutional layers have 192 kernels), while achieving competitive results with BPG.

From the results that were obtained using the aforementioned methods, it can be observed that all aforementioned autoencoder-style compression methods with/without hyperprior outperform conventional compression methods in most cases. The second observation is that the autoencoder-style compression framework with hyperprior (Ballé *et al.*, 2018; Lee *et al.*, 2018) can give a better compression performance than an autoencoder-style compression framework with no hyperprior (Theis *et al.*, 2017) in terms of both PSNR and MS-SSIM. The hyper-encoder/decoder network will however lead to an increase in computational complexity and encoding/decoding time. But with the improvement in computational power in recent years, the autoencoder-style compression framework with hyperprior is a competitive framework and was adopted in the exploration studies of the JPEG AI standard (Ascenso *et al.*, 2019).

4.5.3 Autoencoder-based Codecs with Hyperprior and with Additional Enhancement Modules

Table 4.6 provides a summary of the experimental setup and obtained performance results using the autoencoder-based approaches with hyperprior and some extra modules as described in Section 2.2.1.

The EDIC model that was proposed by Liu *et al.* (2020b) was trained using a set of 256×256 patches randomly cropped from 20,745 high-quality images from Flickr (Flickr, 2022). The Kodak dataset was used for testing. Using the PSNR as image quality metric, EDIC gave comparable results to the ones obtained using the methods of Minnen *et al.* (2018) and Lee *et al.* (2018) at low bitrates, and better results at high bitrates. Using the MS-SSIM quality metric, EDIC exhibited a better compression performance than BPG, JPEG, JPEG2000 and the method of Ballé *et al.* (2018). At low bitrates, EDIC was comparable with the method of Minnen *et al.* (2018) and worse than the method of

Table 4.6: Performance analysis of autoencoder-based codecs with hyperprior and with additional enhancement modules.

Method	Training Dataset	Testing Dataset	Anchor	Metric	Results Description
EDIC (Liu *et al.*, 2020b)	a set of 256×256 patches randomly cropped from 20,745 high-quality images from Flickr (Flickr, 2022)	Kodak	BPG, JPEG, JPEG 2000, Ballé *et al.* (2018), Minnen *et al.* (2018), Lee *et al.* (2018)	rate-distortion in terms of MS-SSIM	at low bitrates, EDIC was comparable to Minnen *et al.* (2018)'s method, worse than Lee *et al.* (2018)'s method, and better than remaining anchor codecs, while at high bitrates, EDIC outperformed all the anchors.
Cheng *et al.* (2020)	a subset of ImageNet dataset	Kodak dataset and the CLIC professional validation dataset	VVC-intra (VTM 5.2), JPEG, JPEG2000, HEVC-intra, Minnen *et al.* (2018), Lee *et al.* (2018), Ballé *et al.* (2018), and Li *et al.* (2018b)	rate-distortion in terms of PSNR, MS-SSIM	in terms of PSNR, the models optimized for MSE yield competitive results as compared to VVC-intra and better results as compared to other anchors, and in terms of MS-SSIM, the models optimized for MS-SSIM outperforms all the anchors
JointIQ-Net (Lee *et al.*, 2019)	a set of 51,140 256×256 patches extracted from CLIC training set	Kodak	VVC Intra (VTM 7.1), BPG, JPEG2000, Minnen *et al.* (2018), Lee *et al.* (2018), and Ballé *et al.* (2018)	rate-distortion in terms of PSNR, MS-SSIM	compression gains in terms of DB rate reduction of 1.65%(48.40%), 16.96%(14.83%), 26.58%(26.65%), 22.57%(57.35%), and 45.48%(73.65%) using PSNR (MS-SSIM) as the visual quality metrics, as compared to VVC Intra (VTM 7.1), BPF, JPEG2000, method of Lee *et al.* (2018) and method of Ballé *et al.* (2018), respectively.

Continued.

Table 4.6: Continued.

Method	Training Dataset	Testing Dataset	Anchor	Metric	Results Description
Liu *et al.* (2019)	COCO and CLIC training datasets	Kodak dataset	BPG, JPEG and JPEG2000, Ballé *et al.* (2018), Mentzer *et al.* (2018) and Rippel and Bourdev (2017)	rate-distortion in terms of MS-SSIM	yields better compression efficiency as compared to conventional and learning-based compression methods in terms of MS-SSIM
Minnen and Singh (2020)	-	Kodak dataset	BPG, JPEG2000, WebP and JPEG, Minnen *et al.* (2018) and Lee *et al.* (2018)	rate-distortion in terms of PSNR	outperforms both selected conventional and learning-based compression anchors in terms of PSNR
Cai *et al.* (2019)	CLIC dataset	CLIC validation dataset	Ballé *et al.* (2018)	rate-distortion in terms of MS-SSIM	achieves fast encoding/decoding time while preserving high reconstructuion visual qualtiy in terms of MS-SSIM

Lee *et al.* (2018) while at high bitrates, EDIC outperformed all these codecs. The EDIC framework was also trained and tested for video compression. The network in Cheng *et al.* (2020) was trained on a subset of the ImageNet dataset and tested on the Kodak dataset and the CLIC professional validation dataset. For the Kodak dataset, the proposed framework optimized for MSE yields competitive results as compared to VVC-intra (VTM 5.2) and achieves a better performance as compared to other compression methods, including JPEG, JPEG2000, HEVC-intra, and the learning-based compression methods from Minnen *et al.* (2018), Lee *et al.* (2018), Ballé *et al.* (2018), and Li *et al.* (2018b), in terms of PSNR. The framework optimized for MS-SSIM outperformed all the aforementioned compression algorithms in terms of MS-SSIM. For the CLIC professional validation dataset, the proposed networks (optimized on MSE and MS-SSIM) were only compared with the conventional compression methods and the learning-based method of Lee *et al.* (2018). In terms of PSNR, the proposed model optimized for MSE outperformed all other codecs except VVC, while in terms of MS-SSIM, the proposed model optimized for MS-SSIM outperformed all these other codecs.

JointIQ-Net (Lee *et al.*, 2019) was trained on a set of 51,140 256×256 patches extracted from the CLIC training set and was tested on the Kodak dataset. Compared with VVC Intra (VTM 7.1), BPG, JPEG2000, and the learning-based compression frameworks of Minnen *et al.* (2018), Lee *et al.* (2018), and Ballé *et al.* (2018) JointIQ-Net gave a better visual quality in terms of both PSNR and MS-SSIM over all different bitrates. The compression gains expressed in terms of reduction in BD-rate and using PSNR (MS-SSIM) were 1.65%(48.40%), 16.96%(14.83%), 26.58%(26.65%), 22.57%(57.35%), and 45.48%(73.65%) as compared to VVC Intra (VTM 7.1), method of Lee *et al.* (2018), method of Ballé *et al.* (2018), BPG and JPEG2000, respectively. The model of of Liu *et al.* (2019) was trained using the COCO (Lin *et al.*, 2014) and CLIC training datasets and was tested on the Kodak dataset. The obtained experimental results showed that the proposed framework exhibits a better compression efficiency as compared to conventional compression methods, like BPG, JPEG and JPEG2000, as well as some other learning-based compression methods (Ballé *et al.*, 2018; Mentzer *et al.*, 2018; Rippel and Bourdev, 2017) in terms of MS-SSIM.

The full model (10 channel-conditioning (CC) slices + latent residual prediction (LRP) + round-based training (training with rounded latent values)) proposed in Minnen and Singh (2020) outperformed conventional compression methods, such as BPG, JPEG2000, WebP and JPEG, and learning-based methods, including the work of Minnen *et al.* (2018) and Lee *et al.* (2018), when tested on the Kodak dataset in terms of PSNR. Experiments also showed that the RD performance increases with more CC slices. The model proposed in Cai *et al.* (2019), which was trained on the CLIC dataset and evaluated on the CLIC validation dataset, was able to achieve fast encoding/decoding time while still preserving a high reconstruction visual quality in terms of MS-SSIM.

Figures 4.2 and 4.3 show the rate-distortion (RD)-curve of the aforementioned autoencoder-based compression methods tested on the Kodak dataset[2]. The rate in Figures 4.2 and 4.3 is measured using bitrate per pixel (bpp) while the distortion is measured using PSNR and MS-SSIM, respectively. In both figures, the conventional codecs are shown in dashdot lines, the autoencoder-based compression (with/without hyperprior) methods are shown in solid lines, and the learning-based methods that were further developed by adding extra modules are shown in dashed lines. From both figures, we can see that, overall, the learning-based compression methods with hyperprior (Ballé *et al.*, 2018; Minnen *et al.*, 2018; Lee *et al.*, 2018; Lin *et al.*, 2020a; Ladune *et al.*, 2020) outperform the conventional ones (JPEG, BPG, and JPEG2000) and by incorporating a channel attention module and a decoder-side enhancement network (Liu *et al.*, 2020b), two attention modules added in the middle and at the end of the encoder and decoder network (Cheng *et al.*, 2020), model parameter estimator (Lee *et al.*, 2019), conditional context module (Liu *et al.*, 2019), channel conditioning and latent residual prediction (Minnen and Singh, 2020) and sub-pixel layer (Cai *et al.*, 2019), the autoencoder-based compression methods can achieve even further improved RD-curve performance.

[2]Data points in the figures are extracted from curves in the original papers.

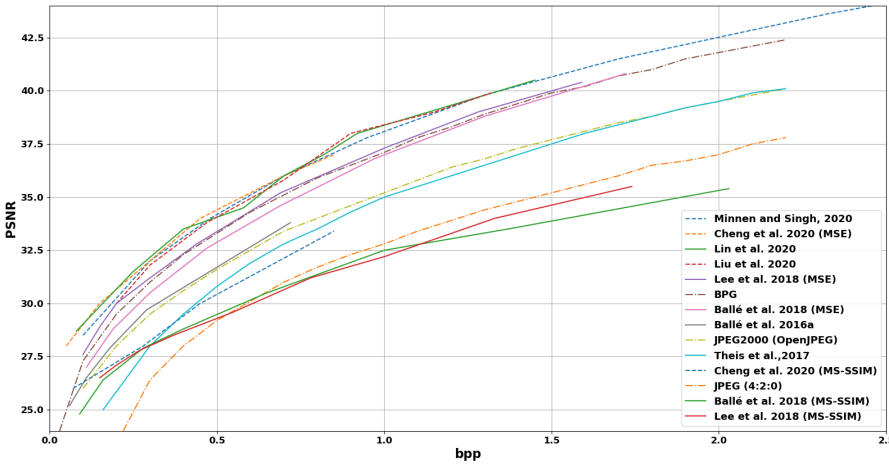

Figure 4.2: PSNR results for different autoencoder-based compression methods on the Kodak dataset.

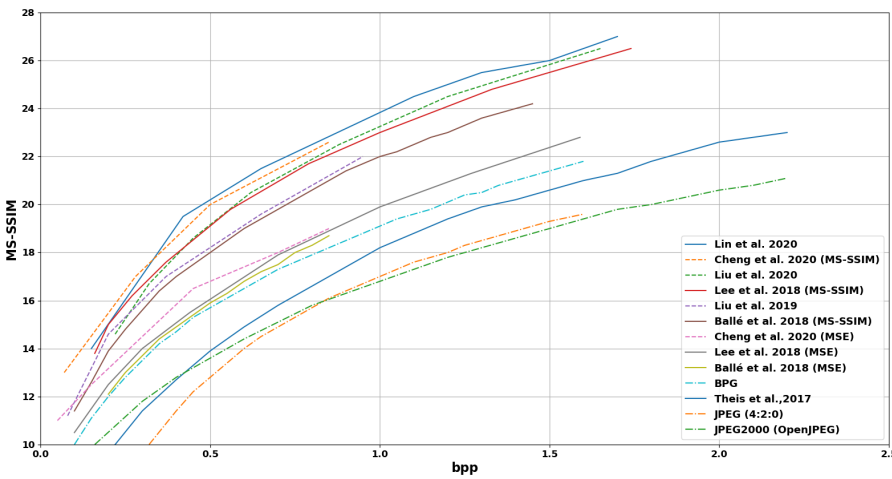

Figure 4.3: MS-SSIM results for different autoencoder-based compression methods on the Kodak dataset.

4.5.4 Autoencoder-based Video Compression Methods

Table 4.7 provides a summary of the experimental setup and obtained performance results of the autoencoder-based video compression methods as described in Section 2.2.1.

Table 4.7: Performance analysis of autoencoder-based video compression methods.

Method	Training Dataset	Testing Dataset	Anchor	Metric	Results Description
DVC (Lu *et al.*, 2019a)	Vimeo-90k (Xue *et al.*, 2019)	UVG (Mercat *et al.*, 2020) and HEVC dataset (Sullivan *et al.*, 2012) (Class B, C, D, and E)	H.264 (Wiegand *et al.*, 2003), H.265 (Sullivan *et al.*, 2012)	PSNR, MS-SSIM	outperform H.264 in in terms of PSNR and MS-SSIM and achieve similar performance with H.265 in terms of MS-SSIM.
M-LVC (Lin *et al.*, 2020b)	Vimeo-90k (Xue *et al.*, 2019)	UVG (Mercat *et al.*, 2020) and HEVC dataset (Sullivan *et al.*, 2012) (Class B, C, D, and E)	H.264 (Wiegand *et al.*, 2003), H.265 (Sullivan *et al.*, 2012), DVC (Lu *et al.*, 2019a), Wu *et al.* (2018)	PSNR, MS-SSIM	outperform DVC on both dataset in terms of PSNR, show better performance compared to H.265 in terms of both PSNR and MS-SSIM.
Lu *et al.* (2020)	Vimeo-90k (Xue *et al.*, 2019)	HEVC dataset (Sullivan *et al.*, 2012) (Class B, C, and D), VTL (VTL, 2022), UVG (Mercat *et al.*, 2020), and MCL-JVC (Wang *et al.*, 2016)	H.264 (Wiegand *et al.*, 2003), H.265 (Sullivan *et al.*, 2012), DVC (Lu *et al.*, 2019a), Wu *et al.* (2018), Djelouah *et al.* (2019)	PSNR, MS-SSIM	outperform H.264/H.265, and DVC interms of PSNR and MS-SSIM, can also achieve competitive or better results compared to Wu *et al.* (2018) and Djelouah *et al.* (2019).
					Continued.

Table 4.7: Continued.

Method	Training Dataset	Testing Dataset	Anchor	Metric	Results Description
FVC (Hu *et al.*, 2021)	Vimeo-90k (Xue *et al.*, 2019)	HEVC dataset (Sullivan *et al.*, 2012) (Class B, C, and D), VTL (VTL, 2022), UVG (Mercat *et al.*, 2020), and MCL-JVC (Wang *et al.*, 2016)	H.265 (Sullivan *et al.*, 2012), DVC (Lu *et al.*, 2019a), Lu *et al.* (2020), Djelouah *et al.* (2019), and Agustsson *et al.* (2020)	PSNR, MS-SSIM	outperforms the anchors on all test datasets in terms of PSNR and MS-SSIM.
DeepPVC-net (Park and Kim, 2021)	UGC (UGC, 2022)	HEVC dataset (Sullivan *et al.*, 2012) (Class B, C, D and E) and UVG (Mercat *et al.*, 2020)	H.264 (Wiegand *et al.*, 2003), H.265 (Sullivan *et al.*, 2012), DVC (Lu *et al.*, 2019a), M-LVC (Lin *et al.*, 2020b), Wu *et al.* (2018), Agustsson *et al.* (2020), Habibian *et al.* (2019), and Yang *et al.* (2020)	MS-SSIM	outperforms all the anchor methods over most of the bitrate range in terms of MS-SSIM.

The DVC (Lu *et al.*, 2019a) video compression framework is trained on the Vimeo-90k (Xue *et al.*, 2019) dataset and tested on the UVG dataset (Mercat *et al.*, 2020) and the HEVC Common Testing Sequences (Sullivan *et al.*, 2012) (Class B, Class C, Class D, and Class E). The proposed DVC framework outperforms H.264 (Wiegand *et al.*, 2003) on both the HEVC dataset and the UVG dataset in terms of both PSNR and MS-SSIM and achieves similar performance with H.265 (Sullivan *et al.*, 2012) in terms of MS-SSIM. M-LVC proposed by Lin *et al.* (2020b) is also trained on the Vimeo-90 dataset and tested on the UVG and HEVC dataset. By using multiple previous frames to generate motion vectors, M-LVC outperforms DVC on both datasets by a large margin in terms of PSNR. M-LCV also shows better compression performance compared to H.265 in terms of both PSNR and MS-SSIM. The video compression framework proposed in Lu *et al.* (2020) is also trained on the Vimeo-90k dataset. Different from previous works, Lu *et al.* (2020) test their proposed methods on 4 different datasets: HEVC (Calss B, C, and D), VTL, UVG, and MCL-JVC. This framework can achieve a better compression performance as compared to H.264, H.265, and DVC on all datasets in terms of both PSNR and MS-SSIM. The authors also compared their work to two state-of-the-art frame interpolation methods (Wu *et al.*, 2018; Djelouah *et al.*, 2019). Experiments show that the proposed methods can achieve competitive, and sometimes even better, results than existing methods.

FVC (Hu *et al.*, 2021) is trained on the Vimeo-90k (Xue *et al.*, 2019) and tested on the HEVC (Calss B, C, D and E), VTL, UVG, and MCL-JVC datasets. Compared to H.265 (Sullivan *et al.*, 2012), FVC can result in up to 18% reduction in bitrate on average over all test datasets. Experiments also show that FVC outperforms DVC (Lu *et al.*, 2019a), and the methods of Lu *et al.* (2020), Djelouah *et al.* (2019), and Agustsson *et al.* (2020). DeepPVCnet (Park and Kim, 2021) is trained on the YouTube UGC dataset (UGC, 2022) and tested on the HEVC Common Testing Sequences (Sullivan *et al.*, 2012) (Class B, C, D and E) and UVG dataset (Mercat *et al.*, 2020). Results show that the proposed DeepPVCnet outperforms all the anchor methods over most of the bitrate range in terms of MS-SSIM. Park and Kim (2021) also give BD-rate values with the anchor of H.264 for each sequence in

HEVC Common Testing Sequences. DeepPVCnet can achieve better compression performance over H.264 and H.265 over most of the test sequences.

4.6 Generative Compression Methods

Table 4.8 provides a summary of the experimental setup and performance results that are obtained when using the generative compression frameworks as described in Section 2.2.2.

The generative compression framework of Rippel and Bourdev (2017) was trained on 128×128 patches sampled randomly from the Yahoo Flickr Creative Common 1000 Million dataset (Thomee *et al.*, 2016) and was tested on the Kodak as well as the RAISE-1k dataset. For both testing sets, the proposed algorithm is performed in the RGB as well as the YCbCr color space. The trained model's performance was compared with conventional compression methods, such as JPEG, JPEG2000, WebP, as well as more recent learning-based compression methods (such as Toderici *et al.*, 2017; Theis *et al.*, 2017; Ballé *et al.*, 2016a; Johnston *et al.*, 2018, in terms of 1) bitrate (bpp), 2) MS-SSIM and 3) computation time for encoding and decoding. The results show that for both datasets, in both color spaces, the proposed algorithm achieves the highest MS-SSIM over all bitrates. The GC models, for compression of diverse natural images of Agustsson *et al.* (2019), were trained using 188,000 images from the *Open Images* dataset (Krasin *et al.*, 2017) and evaluated on the Kodak dataset as well as 20 randomly selected images from the RAISE1K dataset. The GC models produce images with much finer details than BPG while BPG uses 95% and 124% more bits than the proposed models for Kodak and RAISE1K, respectively.

In Kudo *et al.* (2019), the proposed network aims to maximize mutual information between the coding features and the reconstructed images while still preserving a good perceived naturalness. The dataset used is the human face data set CelebA (Liu *et al.*, 2015). The dataset is divided into 200,000 training data elements and 25 test data elements. Results show that compared with BPG and the method of Agustsson *et al.* (2019), the proposed network gives a better performance in terms

Table 4.8: Performance analysis of generative compression methods.

Method	Training Dataset	Testing Dataset	Anchor	Metric	Results Description
Rippel and Bourdev (2017)	128×128 patches sampled randomly from the YFCC100m (Thomee *et al.*, 2016)	Kodak and RAISE-1k dataset	JPEG, JPEG2000, WebP, Toderici *et al.* (2017), Theis *et al.* (2017), Ballé *et al.* (2016a), and Johnston *et al.* (2018)	bpp, MS-SSIM, computation time for encoding and decoding	achieves the highest MS-SSIM and the smallest compression size over all bitrates
GC models (Agustsson *et al.*, 2019)	188k images from the *Open Images* dataset (Krasin *et al.*, 2017)	Kodak and 20 randomly selected images from the RAISE-1k dataset	BPG	subjective metric, bitrate	the GC models produce images with much finer details than BPG while BPG uses 95% and 124% more bits than the proposed models for Kodak and RAISE1K, respectively.
Kudo *et al.* (2019)	200,000 data elements from CelebA (Liu *et al.*, 2015)	25 data elements from CelebA	BPG and (Agustsson *et al.*, 2019)	subjective evaluation scores for naturalness and similarity	achieves higher subjective scores averaged over all testing data
NCode (Santurkar *et al.*, 2018)	CelebA (Liu *et al.*, 2015), UT Zappos50K (Yu and Grauman, 2014) and MIT Places (outdoor scenes) (Zhou *et al.*, 2014) datasets	CelebA (Liu *et al.*, 2015), UT Zappos50K (Yu and Grauman, 2014) and MIT Places (outdoor scenes) (Zhou *et al.*, 2014) datasets	JPEG, JPEG2000 (Toderici *et al.*, 2015)	SSIM, subjective visual quality assessment	yields higher reconstruction quality in terms of SSIM and subjective visual quality assessment as compared to the anchors

of subjective evaluation scores for naturalness and similarity to the original uncompressed images. "Naturalness" represents the naturalness of the image when viewed independently and "similarity" represents the similarity when compared with the original reference images. Both evaluation scores were averaged over 10 image experts.

The NCode model proposed in (Santurkar *et al.*, 2018) uses the CelebA (Liu *et al.*, 2015), UT Zappos50K (Yu and Grauman, 2014) and MIT Places (outdoor scenes) (Zhou *et al.*, 2014) datasets for compression benchmarks. Results show that the NCode model yields higher reconstruction quality, in terms of SSIM and subjective visual quality assessment, as compared to JPEG, JPEG2000, and the thumbnail compression approach of Toderici *et al.* (2015).

For end-to-end learning-based compression approaches, including autoencoder-based and generative compression, the advantage is that most of these methods outperform the conventional compression algorithms. Furthermore, by training on specific datasets, the framework can be further optimized for specific image types and visual applications.

But there are still some drawbacks for these methods. Due to the complexity of deep neural networks, the encoding/decoding procedures can be very time consuming. Most of these methods require a GPU. Compared with conventional compression methods, the hardware requirement is one main reason that is preventing such methods to be deployed for real-time image compression on power-constrained mobile devices. Another disadvantage is that separate compression models need to be trained and saved for different compression bitrates, which will further enlarge the memory requirements of learning-based compression methods.

5

Recent Learning-Based Visual Compression Standardization Efforts

As discussed in Sections 2 and 4, learning-based visual compression methods have been shown to achieve competitive, or even better, compression efficiency, in terms of both objective quality metrics and subjective visual quality assessment as compared to the conventional compression methods. Given the fact that most learning-based visual compression methods suffer from relatively high computational power and long encoding/decoding time, adopting learning-based methods as part of image compression standardization activities is still an ongoing and challenging work. So far, the presented learning-based compression methods mainly aim at maximizing the visual image quality as perceived by a human for a range of bitrates. With the significant increase of applications incorporating automated visual tasks such as image classification and segmentation, many images are captured, stored and/or transmitted not only to be viewed by a human but also to be used for image processing and machine vision tasks. To this end, compression methods and standards are currently being developed targeting effective performance for both human visualization and for automated image processing and computer vision tasks. As such there is a growing interest in developing compression algorithms that can achieve high compression efficiency while benefiting visual tasks.

5.1 Task-Driven Compression Algorithms

In many cases images are captured for certain visual tasks, *e.g.*, images captured by autonomous-driving cars for segmentation and object detection. This led researchers to explore learning-based compression methods that not only can improve the compression efficiency but can also improve the performance of the visual task. Such task-driven compression algorithms aim to improve the compression performance by making the compression algorithm target certain visual tasks. He *et al.* (2019) proposed a learning-based Semantically Structured Coding (SSC) framework to generate a Semantically Structured Bit-stream (SSB). The generated Semantically Structured Bit-stream (SSB) has each of its parts corresponding to a certain object and can thus be directly used for detection. In Krishnaraj *et al.* (2020), the authors are concerned with real-time compression in the context of Internet of Underwater Things (IoUT). They proposed a DWT-CNN architecture comprised of two CNNs, C-CNN and R-CNN. The C-CNN compacts the original image of IoUT objects into a compact representation while the R-CNN reconstructs the decoded compact image into the original size. The compact representation is transformed, compressed, and decoded with a DWT-based image codec.

Li and Ji (2020) proposed a novel end-to-end Neural Image Compression and Explanation (NICE) framework that can produce a classifier prediction, a sparse mask of the input image which indicates the salient regions for the classifier, and a mixed-resolution image which can be compressed efficiently. The purpose of this model is to compress the input image to minimal size while preserving the classification accuracy. To achieve this goal, the authors proposed to use a mask generator on the input image to generate a sparse mask that indicates the salient regions in the image. Then the input image is transformed into a mixed-resolution image in which the salient regions are represented at a relatively high resolution as compared to the non-salient background region which is represented at a lower resolution. Compression of the mixed-resolution image can achieve a higher compression rate as compared to compressing the original input image while retaining similar classification accuracy.

5.2 JPEG-AI

From the task-driven compression algorithms introduced in Section 5.1, it can be seen that most of these algorithms have a framework in which the visual tasks are performed on the fully/partially decoded images. Almost all the learning-based compression methods are implemented through DNNs or CNNs, and to achieve better compression performance, deeper and more complicated networks are being used in such compression methods. One problem with such compression methods is that they may require high computing power as well as long encoding/decoding time. As mentioned before, one reason for task-driven compression research is that many images are captured for visual task purposes. Based on the ability of neural networks to keep important information of the original image in the generated feature map, researchers proposed the idea of performing visual tasks directly on the quantized feature map. Through this approach, the transmitted bitstream need only to be entropy decoded without having to fully decode the image, which can result in significant computational savings.

The JPEG-AI group, as part of Working Group 1 (WG1/JPEG) of ISO/IEC JTC 1/SC29 (Coding of audio, picture, multimedia and hypermedia information standardization committee), is working to create a new learning-based image coding standard which not only can achieve a significant compression efficiency improvement while optimizing the perceived visual quality for humans, but also offers a single-stream compact compressed-domain representation which can be used directly for visual tasks. The JPEG AI project is a joint standardization effort between ISO/IEC JTC1/SC29/WG1 and ITU-T SG16. Different from conventional compression methods where a full decoding process is needed, this new standardization effort calls for a learning-based compression method that can provide a compressed-domain representation (also called latent representation) which can be directly used for image processing and visual tasks. The latent representation can take the form of quantized feature maps that are produced by a deep neural network.

Figure 5.1 illustrates the framework for the JPEG AI framework (Ascenso and Upenik, 2021; WG1, 2022). As shown in Figure 5.1, the

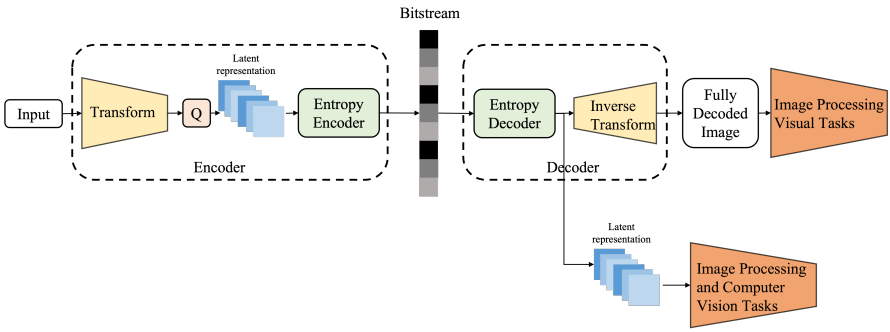

Figure 5.1: JPEG AI learning-based image compression framework.

input image is transformed and quantized to generate a latent representation, also known as compressed-domain representation, which will later be entropy encoded and transmitted from the encoder side to the decoder side. At the decoder side, the bitstream will be entropy decoded and used to reconstruct the output image using an inverse transform. Instead of performing a pixel-domain visual task on the reconstructed images, visual tasks can be performed directly on the entropy-decoded latent representation since the latent representation contains the necessary information not only for standard reconstruction, but also for different image processing algorithms, such as super-resolution, denoising, color correction, and inpainting, and for different computer vision tasks, such as image classification, object detection, segmentation, and action recognition.

5.3 JPEG Pleno Point Cloud

A point cloud is a set of data points in a space. Different from images or video frames which are spatial 2D data, point cloud data is always used to represent a 3D shape or 3D object. Each point has one set of Cartesian coordinates to show its spatial position while it can still has values to represent its property like RGB color intensity, gray-scale intensity, etc. Due to the special structure and large amount of data, the compression for point cloud data is a research area different from conventional image/video compression. Researchers have been working on seeking efficient compression standards for point cloud.

Most point cloud data are gathered through 3D scanning equipments like radar (2D/3D), stereo camera, and time-of-light camera while they are widely used in multiple ways such as autonomous driving, virtual reality, augmented reality, 3D printing, manufacturing, etc. Similar to the JPEG-AI group, the JPEG Pleno Point Cloud group aims to create a learning-based coding standard for point clouds which can offer a single-stream, compact compressed-domain representation and support flexible data access functionalities (Perry, 2021). The expected new learning-based compression method should be able to satisfy not only the need for human visualization and interaction, but also the need for machines to perform both 3D processing and computer vision tasks in the compressed-domain.

One main goal of 3D point cloud compression methods is to reduce the redundancy while preserving necessary information (Cao *et al.*, 2019). Based on the dimension among which the redundancy is reduced, Cao *et al.* (2019) classified the methods into 3 categories: 1D traversal, 2D projection, and 3D decorrelation. 1D traversal compression methods employ a construction of tree-based connectivity to exploit the neighborhood relations between data points, thus converting the geometry data into a 1D signal (Gumhold *et al.*, 2005; Merry *et al.*, 2006). One main drawback of 1D traversal compression methods is that it does not sufficiently take into consideration the 3D spatial correlation. 2D projection methods are seeking projection/mapping algorithms to convert 3D cloud point data into a 2D image/video and then compress the resulting 2D image/video using existing 2D image/video compression methods (Ochotta and Saupe, 2004; Pauly and Gross, 2001). 3D decorrelation methods are the most exploited methods and usually achieve better compression performance than the previous two. Frameworks of 3D decorrelation methods use tree-based representations (Huang *et al.*, 2006; Garcia and Queiroz, 2018; Kathariya *et al.*, 2018), including octree, binary tree and kd-tree, LOD (Level Of Detail) - based methods (Fan *et al.*, 2013; Kitago and Gopi, 2006), clustering-based methods (Zhang *et al.*, 2018), and transform-based methods, including Region-Adaptive Hierarchical Transforms (De Queiroz and Chou, 2016), Graph Fourier Transforms (Thanou *et al.*, 2016), and Gaussian Process Transforms (De Queiroz and Chou, 2017).

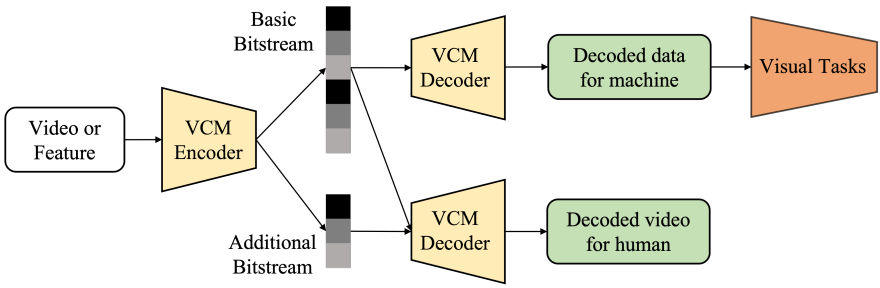

Figure 5.2: Illustration for video coding for machine scope.

5.4 Video Coding for Machine (VCM)

The Moving Picture Experts Group (MPEG) is also working on developing learning-based video coding for machine through leveraging learning-based methods. Different from traditional video coding methods which are optimized for visual quality from a human perspective, video coding for machine focuses on optimization for multiple machine vision tasks such as object detection, object segmentation and object tracking.

The targeted Video Coding for Machine compression methods aim to generate a bitstream through compressing either extracted features or a video stream (MPEG, 2020).

MPEG launched a standardization effort to standardize a video coding scheme which can generate a compressed bitstream with sufficiently smaller size than the state-of-art video compression methods (e.g., HEVC and VVC) such that the decompressed bitstream can result in decoded data optimized for machine vision tasks (e.g., classification, segmentation, object detection and tracking). Since machines "see" the data in a different way than a human, in order to support the generation of video data for human consumption, an additional bitstream can be sent as part of the MPEG VCM framework (Figure 5.2). As shown in Figure 5.2, two bitstreams are generated by the VCM encoder, a basic bitstream and an additional bitstream. The basic bitstream is transmitted to the decoder to generate decoded data for machine vision. The additional bitstream will only be transmitted when a video for human is needed at the decoder end.

Duan *et al.* (2020) presented a review of state-of-the-art video compression and feature compression methods and proposed a VCM architecture that shows the advantage of collaborative compression in enhancing the performance in terms of video and feature coding efficiency and improved performance of both human and machine visual tasks. In Duan *et al.* (2020), three potential VCM architectures are tested: *Deep Intermediate Feature Compression*, *Predictive Coding with Collaborative Feedback*, and *Enhancing Predictive Coding with Scalable Feedback*. Based on the work of Chen *et al.* (2019), the *Deep Intermediate Feature Compression* approach takes the intermediate layer features from a deep framework, compressed and transmitted for human/machine visual tasks. Adopting the pipeline for joint feature/video compression approach of Xia *et al.* (2020) and Hu *et al.* (2020), the *Predictive Coding With Collaborative Feedback* approach first compresses the selected key frames with conventional compression methods, and then transmits these as side information to the decoder for the decompression and reconstruction of other non-key frames.

To help with the reconstruction of non-key frames, a sparse point prediction network is employed to extract the key frames, which will be later compressed through a feature compression methods and transmitted to the decoder to help with non-key frame reconstruction and video analysis. Based on the previous framework, the *Enhancing Predictive Coding with Scalable Feedback* approach uses an enhanced architecture by adding a scalable feedback, which allows the network to adaptively launch the feedback to help further improve the compression performance. Fischer *et al.* (2020) proposed the feature-based rate-distortion optimization (FRDO) which aims to improve the coding performance while the decoded frame is used for DNN-based visual tasks. To accomplish this, the authors replace the pixel-domain distortion metric in VTM-8.0 with a distortion metric calculated from the feature map that is generated by the first layer (i.e., the first convolution, ReLU, and pooling layers) of an employed trained classification or detection network. Based on the block partitioning in VVC, the proposed network also takes Coding Units (CUs) as input instead of the whole frame.

6

Conclusion and Future Directions

Learning-based visual compression is a growing area of research and has been the focus of recent ISO/IEC and ITU-T standardization activities. Recent learning based compression frameworks have demonstrated a higher compression efficiency in terms of their ability to achieve a lower bitrate and/or a higher visual quality of the decompressed visual media as compared to popular conventional compression methods. This work summarizes developments in this area over the past decade and presents a comprehensive review of noteworthy learning-based compression models. These models span a range of models from the earliest attempts in trying to embed learning-based modules into conventional compression methods to more recent end-to-end learning-based approaches. This work also presents common datasets that are used for training and testing the learning-based compression models as well as measures for compression performance assessment. Recent standardization efforts with a focus on learning-based image and video coding are described along with the adoption of learning-based approaches to enable more effective task-driven compression schemes.

Future directions for enabling the adoption of learning-based codecs for real-time image and video compression and transmission applications

include devising low-complexity and low-delay learning-based methods coupled with the capability for online incremental learning and fast adaptation.

References

Adams, M. (2006). "JasPer Software Reference Manual (Version 1.900.0". *ISO/IEC JTC 1/SC 29/WG 1, N 2415*. Dec. URL: https://www.ece.uvic.ca/~frodo/jasper/.

Agustsson, E., D. Minnen, N. Johnston, J. Balle, S. J. Hwang, and G. Toderici. (2020). "Scale-Space Flow for End-to-End Optimized Video Compression". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8503–8512.

Agustsson, E. and R. Timofte. (2017). "NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and study". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 126–135.

Agustsson, E., M. Tschannen, F. Mentzer, R. Timofte, and L. V. Gool. (2019). "Generative Adversarial Networks for Extreme Learned Image Compression". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 221–231.

Akbari, M., J. Liang, and J. Han. (2019). "DSSLIC: Deep Semantic Segmentation-based Layered Image Compression". In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2042–2046. DOI: 10.1109/ICASSP.2019.8683541.

Amestoy, T., A. Mercat, W. Hamidouche, D. Menard, and C. Bergeron. (2019). "Tunable VVC Frame Partitioning Based on Lightweight Machine Learning". *IEEE Transactions on Image Processing*. 29: 1313–1328.

Andrews, D. F. and C. L. Mallows. (1974). "Scale Mixtures of Normal Distributions". *Journal of the Royal Statistical Society: Series B (Methodological)*. 36(1): 99–102.

Arbeláez, P., M. Maire, C. Fowlkes, and J. Malik. (2011). "Contour Detection and Hierarchical Image Segmentation". *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 33(5): 898–916. DOI: 10.1109/TPAMI.2010.161.

Ascenso, J. and E. Upenik. (2021). "White Paper on JPEG AI Scope and Framework". *ISO/IEC JTC1/SC29/WG1, N90049*. Jan.

Ascenso, J., P. Akyzi, M. Testolina, A. Boev, and E. Alshina. (2019). "Performance Evaluation of Learning based Image Coding Solutions and Quality Metrics". *ISO/IEC JTC 1/SC29/WG1 N85013, 85th JPEG Meeting*. Nov.

Asuni, N. and A. Giachetti. (2014). "TESTIMAGES: a large-scale archive for testing visual devices and basic image processing algorithms". In: *STAG: Smart Tools & Apps for Graphics*. 63–70.

Ballé, J., V. Laparra, and E. P. Simoncelli. (2016a). "END-TO-END OPTIMIZED IMAGE COMPRESSION". *arXiv preprint arXiv:1611.01704*.

Ballé, J., V. Laparra, and E. P. Simoncelli. (2016b). "End-to-end optimization of nonlinear transform codes for perceptual quality". In: *2016 Picture Coding Symposium (PCS)*. 1–5. DOI: 10.1109/PCS.2016.7906310.

Ballé, J., D. Minnen, S. Singh, S. J. Hwang, and N. Johnston. (2018). "VARIATIONAL IMAGE COMPRESSION WITH A SCALE HYPERPRIOR". *arXiv preprint arXiv:1802.01436*.

Bellard, F. (2018). "BPG Image Format (2018)". URL: https://bellard.org/bpg/.

Benjak, M., H. Meuel, T. Laude, and J. Ostermann. (2021). "Enhanced Machine Learning-Based Inter Coding for VVC". In: *2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. IEEE. 021–025.

Bjøntegaard, G. (2001). "VCEG-M33: Calculation of Average PSNR Differences between RD curves". *Video Coding Experts Group (VCEG)*: 1520–9210.

Bossen, F. *et al.* (2013). "Common test conditions and software reference configurations". *JCTVC-L1100*. 12(7).

Boyce, J., K. Suehring, X. Li, and V. Seregin. (2018). "JVET Common Test Conditions and Software Reference Configurations". *Joint Vdieo Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, JVET-J1010-v1*. Apr.

Cai, C., G. Lu, Q. Hu, L. Chen, and Z. Gao. (2019). "Efficient Learning Based Sub-pixel Image Compression". In: *CVPR Workshops*. 4.

Cao, C., M. Preda, and T. Zaharia. (2019). "3D Point Cloud Compression: A Survey". In: *The 24th International Conference on 3D Web Technology*. 1–9.

Carreira, J. and A. Zisserman. (2017). "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6299–6308.

Chen, Z., K. Fan, S. Wang, L. Duan, W. Lin, and A. C. Kot. (2019). "Toward Intelligent Sensing: Intermediate Deep Feature Compression". *IEEE Transactions on Image Processing*. 29: 2230–2243.

Cheng, Z., H. Sun, M. Takeuchi, and J. Katto. (2020). "Learned Image Compression with Discretized Gaussian Mixture Likelihoods and Attention Modules". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7939–7948.

Choi, Y., M. El-Khamy, and J. Lee. (2019). "Variable Rate Deep Image Compression With a Conditional Autoencoder". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3146–3154.

CISCO. (2016). "CISCO VNI Complete Forecast Highlights 2021".

Dai, Y., D. Liu, and F. Wu. (2017). "A Convolutional Neural Network Approach for Post-Processing in HEVC Intra Coding". In: *International Conference on Multimedia Modeling*. Springer. 28–39.

Dang-Nguyen, D.-T., C. Pasquini, V. Conotter, and G. Boato. (2015). "RAISE: A Raw Images Dataset for Digital Image Forensics". In: *Proceedings of the 6th ACM multimedia systems conference*. 219–224.

De Queiroz, R. L. and P. A. Chou. (2016). "Compression of 3D Point Clouds Using a Region-Adaptive Hierarchical Transform". *IEEE Transactions on Image Processing*. 25(8): 3947–3956.

De Queiroz, R. L. and P. A. Chou. (2017). "Transform Coding for Point Clouds Using a Gaussian Process Model". *IEEE Transactions on Image Processing*. 26(7): 3507–3517.

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. (2009). "ImageNet: A Large-Scale Hierarchical Image Database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 248–255. DOI: 10.1109/CVPR.2009.5206848.

Djelouah, A., J. Campos, S. Schaub-Meyer, and C. Schroers. (2019). "Neural Inter-Frame Compression for Video Coding". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6421–6429.

Dong, C., Y. Deng, C. C. Loy, and X. Tang. (2015). "Compression Artifacts Reduction by a Deep Convolutional Network". In: *Proceedings of the IEEE International Conference on Computer Vision*. 576–584.

Dong, C., C. C. Loy, K. He, and X. Tang. (2014). "Learning a Deep Convolutional Network for Image Super-Resolution". In: *European Conference on Computer Vision*. Springer. 184–199.

Duan, L., J. Liu, W. Yang, T. Huang, and W. Gao. (2020). "Video Coding for Machines: A Paradigm of Collaborative Compression and Intelligent Analytics". *IEEE Transactions on Image Processing*. 29: 8680–8695.

Egiazarian, K., J. Astola, N. Ponomarenko, V. Lukin, F. Battisti, and M. Carli. (2006). "New Full-Reference Quality Metrics Based on HVS". In: *Proceedings of the Second International Workshop on Video Processing and Quality Metrics*. Vol. 4.

Fan, Y., H. Sun, J. Katto, J. Ming'E, *et al.* (2020). "A Fast QTMT Partition Decision Strategy for VVC Intra Prediction". *IEEE Access*. 8: 107900–107911.

Fan, Y., Y. Huang, and J. Peng. (2013). "Point Cloud Compression Based on Hierarchical Point Clustering". In: *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*. IEEE. 1–7.

Fatima, A., W. Hussain, and S. Rasool. (2021). "Grey is the new RGB: How good is GAN-based image colorization for image compression?" *Multimedia Tools and Applications*. 80(3): 3775–3791.

Fischer, K., F. Brand, C. Herglotz, and A. Kaup. (2020). "Video Coding for Machines with Feature-Based Rate-Distortion Optimization". In: *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*. IEEE. 1–6.

Flickr. (2022). "Flickr homepage". URL: https://flickr.com.

Garcia, D. C. and R. L. de Queiroz. (2018). "Intra-Frame Context-Based Octree Coding for Point-Cloud Geometry". In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE. 1807–1811.

Geiger, A., P. Lenz, C. Stiller, and R. Urtasun. (2013). "Vision Meets Robotics: The KITTI Dataset". *The International Journal of Robotics Research*. 32(11): 1231–1237.

Google. (2015). "WebP Compression Study". URL: https://developers.google.com/speed/webp/docs/webp_study.

Goyal, V. (2001). "Theoretical Foundations of Transform Coding". *IEEE Signal Processing Magazine*. 18(5): 9–21. DOI: 10.1109/79.952802.

Gumhold, S., Z. Kami, M. Isenburg, and H.-P. Seidel. (2005). "Predictive Point-Cloud Compression". In: *ACM SIGGRAPH 2005 Sketches*. 137–es.

Habibian, A., T. v. Rozendaal, J. M. Tomczak, and T. S. Cohen. (2019). "Video Compression with Rate-Distortion Autoencoders". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7033–7042.

He, T., S. Sun, Z. Guo, and Z. Chen. (2019). "Beyond Coding: Detection-driven Image Compression with Semantically Structured Bit-stream". In: *2019 Picture Coding Symposium (PCS)*. 1–5. DOI: 10.1109/PCS48520.2019.8954525.

Hoang, T. M. and J. Zhou. (2021). "Recent Trending on Learning Based Video Compression: A Survey". *Cognitive Robotics*. 1: 145–158.

Hu, Y., S. Yang, W. Yang, L.-Y. Duan, and J. Liu. (2020). "Towards Coding for Human and Machine Vision: A Scalable Image Coding Approach". In: *2020 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. 1–6.

Hu, Z., G. Lu, and D. Xu. (2021). "FVC: A New Framework Towards Deep Video Compression in Feature Space". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1502–1511.

Huang, G., Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. (2017). "Densely Connected Convolutional Networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4700–4708.

Huang, H., I. Schiopu, and A. Munteanu. (2019). "Deep Learning Based Angular Intra-Prediction for Lossless HEVC Video Coding". In: *2019 Data Compression Conference (DCC)*. IEEE. 579–579.

Huang, Y., J. Peng, C.-C. J. Kuo, and M. Gopi. (2006). "Octree-Based Progressive Geometry Coding of Point Clouds." In: *PBG@ SIGGRAPH*. 103–110.

Iizuka, S., E. Simo-Serra, and H. Ishikawa. (2017). "Globally and Locally Consistent Image Completion". *ACM Transactions on Graphics (ToG)*. 36(4): 1–14.

ITU. (2011). "Recommendation ITU-R BT.601-7: Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios". URL: https://www.itu.int/rec/R-REC-BT.601/.

Johnston, N., D. Vincent, D. Minnen, M. Covell, S. Singh, T. Chinen, S. J. Hwang, J. Shor, and G. Toderici. (2018). "Improved Lossy Image Compression with Priming and Spatially Adaptive Bit Rates for Recurrent Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4385–4393.

Kathariya, B., L. Li, Z. Li, J. Alvarez, and J. Chen. (2018). "Scalable point cloud geometry coding with binary tree embedded quadtree". In: *2018 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. 1–6.

Kay, W., J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, *et al.* (2017). "The Kinetics Human Action Video Dataset". *arXiv preprint arXiv:1705.06950*.

Kingma, D. P. and M. Welling. (2013). "Auto-Encoding Variational Bayes". *arXiv preprint arXiv:1312.6114*.

Kitago, M. and M. Gopi. (2006). "Efficient and Prioritized Point Subsampling for CSRBF Compression." In: *PBG@ SIGGRAPH*. Citeseer. 121–128.

Krasin, I., T. Duerig, N. Alldrin, V. Ferrari, S. Abu-El-Haija, A. Kuznetsova, H. Rom, J. Uijlings, S. Popov, A. Veit, *et al.* (2017). "OpenImages: A public dataset for large-scale multi-label and multi-class image classification". *Dataset available from https://github.com/openimages*. 2(3): 18.

Krishnaraj, N., M. Elhoseny, M. Thenmozhi, M. M. Selim, and K. Shankar. (2020). "Deep learning model for real-time image compression in Internet of Underwater Things (IoUT)". *Journal of Real-Time Image Processing*. 17(6): 2097–2111.

Kudo, S., S. Orihashi, R. Tanida, and A. Shimizu. (2019). "GAN-based Image Compression Using Mutual Information Maximizing Regularization". In: *2019 Picture Coding Symposium (PCS)*. 1–5. DOI: 10.1109/PCS48520.2019.8954548.

Ladune, T., P. Philippe, W. Hamidouche, L. Zhang, and O. Déforges. (2020). "Binary Probability Model for Learning Based Image Compression". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2168–2172. DOI: 10.1109/ICASSP40776.2020.9053997.

Larson, E. C. and D. M. Chandler. (2010). "Most Apparent Distortion: Full-Reference Image Quality Assessment and the Role of Strategy". *Journal of electronic imaging*. 19(1): 011006.

Le Callet, P. and F. Autrusseau. (2005). "Subjective Auality Assessment IRCCyN/IVC Database".

Lee, J., S. Cho, and S.-K. Beack. (2018). "Context-adaptive Entropy Model for End-to-end Optimized Image Compression". In: *International Conference on Learning Representations*.

Lee, J., S. Cho, and M. Kim. (2019). "An End-to-End Joint Learning Scheme of Image compression and Quality Enhancement with Improved Entropy Minimization". *arXiv preprint arXiv:1912.12817*.

Li, J., B. Li, J. Xu, R. Xiong, and W. Gao. (2018a). "Fully Connected Network-based Intra Prediction for Image Coding". *IEEE Transactions on Image Processing*. 27(7): 3236–3247.

Li, M. (2019). "A Better Color Space Conversion Based on Learned Variances for Image Compression". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 5.

Li, M., C. Xia, J. Hu, Z. Huang, Y. Zhang, D. Chen, J. Zan, G. Li, and J. Nie. (2019). "VimicroABCnet: An Image Coder Combining A Better Color Space Conversion Algorithm and A Post Enhancing Networ." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 5.

Li, M., W. Zuo, S. Gu, D. Zhao, and D. Zhang. (2018b). "Learning Convolutional Networks for Content-weighted Image Compression". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3214–3223.

Li, T., M. Xu, R. Tang, Y. Chen, and Q. Xing. (2021). "DeepQTMT: A Deep Learning Approach for Fast QTMT-Vased CU Partition of Intra-Mode VVC". *IEEE Transactions on Image Processing*. 30: 5377–5390.

Li, X. and S. Ji. (2020). "Neural Image Compression and Explanation". *IEEE Access*. 8: 214605–214615. DOI: 10.1109/ACCESS.2020.3041416.

Lin, C., J. Yao, F. Chen, and L. Wang. (2020a). "A Spatial RNN Codec for End-to-End Image Compression". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13269–13277.

Lin, J., D. Liu, H. Li, and F. Wu. (2020b). "M-LVC: Multiple Frames Prediction for Learned Video Compression". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3546–3554.

Lin, T.-Y., M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. (2014). "Microsoft COCO: Common Objects in Context". In: *European conference on computer vision*. Springer. 740–755.

Liu, D., Y. Li, J. Lin, H. Li, and F. Wu. (2020a). "Deep Learning-based Video Coding: A Review and A Case Study". *ACM Computing Surveys (CSUR)*. 53(1): 1–35.

Liu, H., T. Chen, P. Guo, Q. Shen, and Z. Ma. (2019). "Gated Context Model with Embedded Priors for Deep Image Compression". *arXiv preprint arXiv:1902.10480*.

Liu, J., G. Lu, Z. Hu, and D. Xu. (2020b). "A Unified End-to-End Framework for Efficient Deep Image Compression". *arXiv preprint arXiv:2002.03370*.

Liu, Z., P. Luo, X. Wang, and X. Tang. (2015). "Deep Learning Face Attributes in the Wild". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 3730–3738.

Lotter, W., G. Kreiman, and D. Cox. (2016). "Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning". *arXiv preprint arXiv:1605.08104*.

Lu, G., C. Cai, X. Zhang, L. Chen, W. Ouyang, D. Xu, and Z. Gao. (2020). "Content Adaptive and Error Propagation Aware Deep Video Compression". In: *European Conference on Computer Vision*. Springer. 456–472.

Lu, G., W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao. (2019a). "DVC: An End-to-End Deep Video Compression Framework". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11006–11015.

Lu, M., T. Chen, H. Liu, and Z. Ma. (2019b). "Learned Image Restoration for VVC Intra Coding". In: *CVPR Workshops*. 4.

Ma, C., D. Liu, X. Peng, L. Li, and F. Wu. (2019a). "Convolutional Neural Network-based Arithmetic Coding for HEVC Intra-predicted Residues". *IEEE Transactions on Circuits and Systems for Video Technology*. 30(7): 1901–1916.

Ma, C., D. Liu, X. Peng, and F. Wu. (2018). "Convolutional Neural Network-based Arithmetic Coding of DC Coefficients for HEVC Intra Coding". In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE. 1772–1776.

Ma, D., F. Zhang, and D. Bull. (2021). "BVI-DVC: A Training Database for Deep Video Compression". *IEEE Transactions on Multimedia*.

Ma, H., D. Liu, R. Xiong, and F. Wu. (2019b). "A CNN-Based Image Compression Scheme Compatible with JPEG-2000". In: *2019 IEEE International Conference on Image Processing (ICIP)*. 704–708. DOI: 10.1109/ICIP.2019.8803835.

Marpe, D., H. Schwarz, and T. Wiegand. (2003). "Context-based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard". *IEEE Transactions on circuits and systems for video technology*. 13(7): 620–636.

Mentzer, F., E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool. (2018). "Conditional Probability Models for Deep Image Compression". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4394–4402.

Mercat, A., M. Viitanen, and J. Vanne. (2020). "UVG Dataset: 50/120fps 4K Sequences for Video Codec Analysis and Development". In: *Proceedings of the 11th ACM Multimedia Systems Conference*. 297–302.

Merry, B., P. Marais, and J. Gain. (2006). "Compression of Dense and Regular Point Clouds". In: *Proceedings of the 4th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*. 15–20.

Minnen, D., J. Ballé, and G. Toderici. (2018). "Joint Autoregressive and Hierarchical Priors for Learned Image Compression". *Advances in neural information processing systems*. 31.

Minnen, D. and S. Singh. (2020). "Channel-Wise Autoregressive Entropy Models for Learned Image Compression". In: *2020 IEEE International Conference on Image Processing (ICIP)*. 3339–3343. DOI: 10.1109/ICIP40778.2020.9190935.

MPEG. (2020). "Evaluation Framework for Video Coding for Machines". *ISO/IEC JTC1/SC29/WG11, W19366*. Apr.

Nowell, M. (2019). "Cisco VNI Forecast Update".

Ochotta, T. and D. Saupe. (2004). *Compression of Point-Based 3D Models by Shape-Adaptive Wavelet Coding of Multi-Height Fields*.

Ohm, J.-R. and G. J. Sullivan. (2018). "Versatile video coding–towards the next generation of video compression". In: *Picture Coding Symposium*. Vol. 2018.

Park, W.-S. and M. Kim. (2016). "CNN-Based In-Loop Filtering for Coding Efficiency Improvement". In: *2016 IEEE 12th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*. 1–5. DOI: 10.1109/IVMSPW.2016.7528223.

Park, W. and M. Kim. (2021). "Deep Predictive Video Compression Using Mode-Selective Uni- and Bi-Directional Predictions Based on Multi-Frame Hypothesis". *IEEE Access*. 9: 72–85. DOI: 10.1109/ACCESS.2020.3046040.

Pauly, M. and M. Gross. (2001). "Spectral Processing of Point-Sampled Geometry". In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. 379–386.

Perry, S. (2021). "JPEG Pleno Point Cloud - Use Cases and Requirements v1.4". *ISO/IEC JTC1/SC29/WG1 N92015, 92th Meeting, Online*. July.

Ponomarenko, N., F. Silvestri, K. Egiazarian, M. Carli, J. Astola, and V. Lukin. (2007). "On Between-Coefficient Contrast Masking of DCT Basis Functions". In: *Proceedings of the Third International Workshop on Video Processing and Quality Metrics*. Vol. 4. Scottsdale USA.

Rippel, O. and L. Bourdev. (2017). "Real-Time Adaptive Image Compression". In: *International Conference on Machine Learning*. PMLR. 2922–2930.

Rissanen, J. and G. Langdon. (1981). "Universal Modeling and Coding". *IEEE Transactions on Information Theory*. 27(1): 12–23. DOI: 10.1109/TIT.1981.1056282.

Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.* (2015). "ImageNet Large Scale Visual Recognition Challenge". *International journal of computer vision*. 115(3): 211–252.

Santurkar, S., D. Budden, and N. Shavit. (2018). "Generative Compression". In: *2018 Picture Coding Symposium (PCS)*. 258–262. DOI: 10.1109/PCS.2018.8456298.

Schaefer, G. and M. Stich. (2003). "UCID: An Uncompressed Color Image Database". In: *Storage and Retrieval Methods and Applications for Multimedia 2004*. Vol. 5307. SPIE. 472–480.

Schiopu, I., H. Huang, and A. Munteanu. (2019). "CNN-Based Intra-Prediction for Lossless HEVC". *IEEE Transactions on Circuits and Systems for Video Technology*. 30(7): 1816–1828.

Sheikh, H., M. Sabir, and A. Bovik. (2006). "A Statistical Evaluation of Recent Full Reference Image Quality Assessment Algorithms". *IEEE Transactions on Image Processing*. 15(11): 3440–3451. DOI: 10.1109/TIP.2006.881959.

Sheikh, H. (2005). "LIVE image quality assessment database release 2". *http://live.ece.utexas.edu/research/quality*.

Sneyers, J. and P. Wuille. (2016). "FLIF: FREE LOSSLESS IMAGE FORMAT BASED ON MANIAC COMPRESSION". In: *2016 IEEE International Conference on Image Processing (ICIP)*. 66–70. DOI: 10.1109/ICIP.2016.7532320.

Sullivan, G. J., J.-R. Ohm, W.-J. Han, and T. Wiegand. (2012). "Overview of the High Efficiency Video Coding (HEVC) Standard". *IEEE Transactions on Circuits and Systems for Video Technology*. 22(12): 1649–1668. DOI: 10.1109/TCSVT.2012.2221191.

Sun, H., C. Liu, J. Katto, and Y. Fan. (2020). "An Image Compression Framework with Learning-based Filter". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 152–153.

Tao, W., F. Jiang, S. Zhang, J. Ren, W. Shi, W. Zuo, X. Guo, and D. Zhao. (2017). "An End-to-End Compression Framework Based on Convolutional Neural Networks". In: *2017 Data Compression Conference (DCC)*. 463–463. DOI: 10.1109/DCC.2017.54.

Taubman, D. and M. Marcellin. (2002). "JPEG2000: Standard for Interactive Imaging". *Proceedings of the IEEE*. 90(8): 1336–1357. DOI: 10.1109/JPROC.2002.800725.

Thanou, D., P. A. Chou, and P. Frossard. (2016). "Graph-Based Compression of Dynamic 3D Point Cloud Sequences". *IEEE Transactions on Image Processing*. 25(4): 1765–1778.

Theis, L., W. Shi, A. Cunningham, and F. Huszár. (2017). "Lossy Image Compression with Compressive Autoencoders". *arXiv preprint arXiv:1703.00395*.

Thomee, B., D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. (2016). "YFCC100M: The New Data in Multimedia Research". *Communications of the ACM*. 59(2): 64–73.

Toderici, G., S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar. (2015). "Variable Rate Image Compression with Recurrent Neural Networks". *arXiv preprint arXiv:1511.06085*.

Toderici, G., W. Shi, R. Timofte, L. Theis, J. Balle, E. Agustsson, N. Johnston, and F. Mentzer. (2020). "Workshop and Challenge on Learned Image Compression (CLIC2020)". URL: http://www.compression.cc.

Toderici, G., D. Vincent, N. Johnston, S. Jin Hwang, D. Minnen, J. Shor, and M. Covell. (2017). "Full Resolution Image Compression with Recurrent Neural Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5306–5314.

UGC. (2022). "YouTube UGC Dataset". URL: https://media.withyoutube.com/.

Van den Oord, A., N. Kalchbrenner, L. Espeholt, K. Kavukcuoglu, O. Vinyals, and A. Graves. (2016). "Conditional Image Generation with PixelCNN Decoders". In: *Advances in Neural Information Processing Systems*. Vol. 29. 4790–4798.

Van Oord, A., N. Kalchbrenner, and K. Kavukcuoglu. (2016). "Pixel Recurrent Neural Networks". In: *International Conference on Machine Learning*. PMLR. 1747–1756.

VTL. (2022). "VIdeo Trace Library (VTL homepage)". URL: http://trace.eas.asu.edu/yuv/index.html.

Wainwright, M. J. and E. P. Simoncelli. (1999). "Scale Mixtures of Gaussians and the Statistics of Natural Images". In: *Advances in Neural Information Processing Systems (NIPS)*. Vol. 12. 855–861.

Wallace, G. (1992). "The JPEG Still Picture Compression Standard". *IEEE Transactions on Consumer Electronics*. 38(1): xviii–xxxiv. DOI: 10.1109/30.125072.

Wang, H., W. Gan, S. Hu, J. Y. Lin, L. Jin, L. Song, P. Wang, I. Katsavounidis, A. Aaron, and C.-C. J. Kuo. (2016). "MCL-JCV: a JND-based H. 264/AVC Video Quality Assessment Dataset". In: *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE. 1509–1513.

Wang, T., M. Chen, and H. Chao. (2017). "A Novel Deep Learning-Based Method of Improving Coding Efficiency from the Decoder-End for HEVC". In: *2017 Data Compression Conference (DCC)*. 410–419. DOI: 10.1109/DCC.2017.42.

Wang, Z., A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. (2004). "Image Quality Assessment: From Error Visibility to Structural Similarity". *IEEE transactions on image processing*. 13(4): 600–612.

Wang, Z., E. P. Simoncelli, and A. C. Bovik. (2003). "Multiscale Structural Similarity for Image Quality Assessment". In: *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*. Vol. 2. Ieee. 1398–1402.

WG1. (2022). "Final Call for Proposals for JPEG AI". *ISO/IEC JTC 1/SC29/WG1, N100095, 94th JPEG Meeting, Online*. Jan.

Wiegand, T., G. Sullivan, G. Bjontegaard, and A. Luthra. (2003). "Overview of the H.264/AVC Video Coding Standard". *IEEE Transactions on Circuits and Systems for Video Technology*. 13(7): 560–576. DOI: 10.1109/TCSVT.2003.815165.

Wu, C.-Y., N. Singhal, and P. Krahenbuhl. (2018). "Video Compression Through Image Interpolation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 416–431.

Xia, S., K. Liang, W. Yang, L.-Y. Duan, and J. Liu. (2020). "An Emerging Coding Paradigm VCM: A Scalable Coding Approach Beyond Feature and Signal". In: *2020 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. 1–6.

Xue, T., B. Chen, J. Wu, D. Wei, and W. T. Freeman. (2019). "Video Enhancement with Task-Oriented Flow". *International Journal of Computer Vision*. 127(8): 1106–1125.

Yan, N., D. Liu, H. Li, B. Li, L. Li, and F. Wu. (2018). "Convolutional Neural Network-based Fractional-pixel Motion Compensation". *IEEE Transactions on Circuits and Systems for Video Technology*. 29(3): 840–853.

Yang, R., F. Mentzer, L. V. Gool, and R. Timofte. (2020). "Learning for Video Compression with Hierarchical Quality and Recurrent Enhancement". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6628–6637.

Yang, R., M. Xu, T. Liu, Z. Wang, and Z. Guan. (2019). "Enhancing Quality for HEVC Compressed Videos". *IEEE Transactions on Circuits and Systems for Video Technology*. 29(7): 2039–2054. DOI: 10.1109/TCSVT.2018.2867568.

Yim, C. and A. C. Bovik. (2010). "Quality Assessment of Deblocked Images". *IEEE Transactions on Image Processing*. 20(1): 88–98.

Yu, A. and K. Grauman. (2014). "Fine-Grained Visual Comparisons with Local Learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhang, K., W. Zhu, and Y. Xu. (2018). "Hierarchical Segmentation Based Point Cloud Attribute Compression". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 3131–3135.

Zhang, R., J.-Y. Zhu, P. Isola, X. Geng, A. S. Lin, T. Yu, and A. A. Efros. (2017). "Real-Time User-Guided Image Colorization with Learned Deep Priors". *arXiv preprint arXiv:1705.02999*.

Zhao, H., J. Shi, X. Qi, X. Wang, and J. Jia. (2017). "Pyramid Scene Parsing Network". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2881–2890.

Zhou, B., A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. (2014). "Learning Deep Features for Scene Recognition using Places Database". *Advances in Neural Information Processing Systems*. 27.

Zhou, B., H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. (2017). "Scene Parsing through ADE20K Dataset". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 633–641.

Zhou, L., C. Cai, Y. Gao, S. Su, and J. Wu. (2018). "Variational Autoencoder for Low Bit-rate Image Compression". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Zhu, L., S. Kwong, Y. Zhang, S. Wang, and X. Wang. (2019). "Generative Adversarial Network-Based Intra Prediction for Video Coding". *IEEE Transactions on Multimedia*. 22(1): 45–58.