Deep Learning-Based Video Coding: A Review and a Case Study

DONG LIU, YUE LI, JIANPING LIN, HOUQIANG LI, and FENG WU, CAS Key Laboratory of Technology in Geo-Spatial Information Processing and Application System, University of Science and Technology of China

The past decade has witnessed the great success of deep learning in many disciplines, especially in computer vision and image processing. However, deep learning-based video coding remains in its infancy. We review the representative works about using deep learning for image/video coding, an actively developing research area since 2015. We divide the related works into two categories: new coding schemes that are built primarily upon deep networks, and deep network-based coding tools that shall be used within traditional coding schemes. For deep schemes, pixel probability modeling and auto-encoder are the two approaches, that can be viewed as predictive coding and transform coding, respectively. For deep tools, there have been several techniques using deep learning to perform intra-picture prediction, inter-picture prediction, cross-channel prediction, probability distribution prediction, transform, post- or in-loop filtering, down- and up-sampling, as well as encoding optimizations. In the hope of advocating the research of deep learning-based video coding, we present a case study of our developed prototype video codec, Deep Learning Video Coding (DLVC). DLVC features two deep tools that are both based on convolutional neural network (CNN), namely CNN-based in-loop filter and CNN-based block adaptive resolution coding. The source code of DLVC has been released for future research.

CCS Concepts: • **Computing methodologies** → **Image compression**;

Additional Key Words and Phrases: Deep learning, image coding, prediction, transform, video coding

ACM Reference format:
Dong Liu, Yue Li, Jianping Lin, Houqiang Li, and Feng Wu. 2020. Deep Learning-Based Video Coding: A Review and a Case Study. *ACM Comput. Surv.* 53, 1, Article 11 (February 2020), 35 pages.
https://doi.org/10.1145/3368405

1 INTRODUCTION

1.1 Image/Video Coding

Image/video coding usually refers to the computing technology that compresses image/video into binary code (i.e., bits) so as to facilitate storage and transmission. The compression may or may not ensure perfect reconstruction of image/video from the bits, which is termed lossless and lossy coding, respectively. For natural image/video, the compression efficiency of lossless coding is

This work is supported by the Natural Science Foundation of China under Grant 61772483.
Authors' address: D. Liu, Y. Li, J. Lin, H. Li, and F. Wu, CAS Key Laboratory of Technology in Geo-Spatial Information Processing and Application System, University of Science and Technology of China, No. 443 Huangshan Road, Hefei, Anhui Province, China, 230027; emails: dongeliu@ustc.edu.cn, {lytt, ljp105}@mail.ustc.edu.cn, {lihq, fengwu}@ustc.edu.cn.

0360-0300/2020/02-ART11 $15.00
https://doi.org/10.1145/3368405

usually below requirement, so most of the efforts are devoted to lossy coding. Lossy image/video coding solutions are evaluated by two aspects: the first is the compression efficiency, commonly measured by the number of bits (coding rate), the less the better; the second is the incurred loss, commonly measured by the quality of the reconstructed image/video compared with the original image/video, the higher the better.

Image/video coding is a fundamental and enabling technology for computer image processing, computer vision, and visual communication. The research and development of image/video coding can be dated back to as early as 1960s, much earlier than the appearance of modern imaging, image processing, and visual communication systems. As an example, Picture Coding Symposium, a prestigious international forum devoted specifically to advancements in image/video coding, started in the year of 1969. Since then, numerous efforts from both academia and industry have been devoted to this field.

Due to the requirement of interoperability, a series of standards regarding image/video coding have been crafted in the past three decades. In international standardization organizations, ISO/IEC has two experts group namely Joint Photographic Experts Group (JPEG) and Moving Picture Experts Group (MPEG) for standardization of image/video coding technology, while ITU-T has its own Video Coding Experts Group (VCEG). These organizations have published several famous, widely adopted standards, such as JPEG [132], JPEG 2000 [113], H.262 (MPEG-2 Part 2) [126], H.264 (MPEG-4 Part 10 or AVC) [140], H.265 (MPEG-H Part 2 or HEVC) [118], and so on. At present, H.265/HEVC, which was formally published in 2013, represents the state-of-the-art image/video coding technology.

Along with the progress of video technology, especially the popularization of ultra-high definition (UHD) video, there is an urgent requirement to further increase compression efficiency so as to accommodate UHD video in limited storage and limited transmission bandwidth. Thus, after HEVC, MPEG and VCEG form the Joint Video Experts Team (JVET) to explore advanced video coding technology, and the team developed Joint Exploration Model (JEM) for study. Moreover, since the year of 2018, the JVET team has been working on a new video coding standard, informally called Versatile Video Coding (VVC), as the successor of HEVC. It is anticipated that VVC may improve the compression efficiency by saving around 50% bits while maintaining the same quality, especially for UHD video, compared to HEVC. Nonetheless, it is worth noting that the improvement of VVC is probably achieved at the cost of multiplicative encoding/decoding complexity.

1.2 Deep Learning for Image/Video Coding

The past decade has witnessed the emerging and booming of deep learning, a class of techniques that are increasingly adopted in the hope of approaching the ultimate goal of artificial intelligence [63]. Deep learning belongs to machine learning technology, and has the distinction of its computational models, known as deep artificial neural networks or deep networks for short, which are composed of multiple (usually more than three) processing layers, each layer is further composed of multiple simple but non-linear basic computational units. One benefit of such deep networks is believed to be the capacity for processing data with multiple levels of abstraction, and converting data into different kinds of representations. Note that these representations are not manually designed; instead, the deep network including the processing layers is learned from massive data using a general machine learning procedure. Deep learning eliminates the necessity of handcrafted representations, and thus is regarded useful especially for processing natively unstructured data, such as acoustic and visual signal, whilst processing such data has been a longstanding difficulty in the artificial intelligence field.

Specifically for processing image/video, deep learning using convolutional neural network (CNN) has revolutionized the paradigm in computer vision and image processing. In 2012, Krizhevsky et al. [61] designed a 8-layer CNN, which won the image classification challenge by a surprisingly low error rate compared with previous works. In 2014, Girshick et al. [36] promoted the performance of object detection by a significant margin with the proposed regions with CNN features. Also in 2014, Dong et al. [29] proposed a 3-layer CNN for single image super-resolution (SR), which outperforms the previous methods in both reconstruction quality and computational speed. In 2017, Zhang et al. [155] presented a deep CNN for image denoising, and demonstrated that a single CNN model may tackle with several different image restoration tasks including denoising, single-image SR, and compression artifact reduction, while these tasks had been studied separately for a long while.

Witnessing such successful cases, experts cannot help but ask whether deep learning can benefit image/video coding as well. In history, artificial neural network is not strange to the image/video coding community. From 1980s to 1990s, a number of works were conducted on neural network-based image coding [30, 51], but then the networks were shallow and the compression efficiency was not satisfactory. Thanks to the abundance of data, the more and more powerful computing platform, and the development of advanced algorithms, it is now possible to train very deep networks with even more than 1,000 layers [43]. Thus, the exploration of using deep learning for image/video coding is worthy of reconsideration, and indeed has been an actively developing research area since 2015. At present, works have shown promising results, confirmed the feasibility of deep learning-based image/video coding. Nonetheless, this technology is far from mature and calls for much more research and development efforts.

In this article, we aim to provide a comprehensive review of the newest reports about deep learning-based image/video coding, until the end of 2018, as well as to present a case study of our developed prototype video codec namely Deep Learning Video Coding (DLVC), so as to make interested readers aware of the status quo. Readers may also refer to [78] and [93] for recently published review papers about the same theme.

The remainder of this article is organized as follows: Section 1.3 presents some preliminaries. Sections 2 and 3 provide a review of related works about using deep learning for image/video coding. The related works are divided into two categories, and reviewed in the two sections respectively. The first category is deep schemes, i.e., new coding schemes that are built primarily upon deep networks; the second category is deep tools, i.e., deep network-based coding tools that are embedded into traditional, non-deep coding schemes; a deep tool may either replace its counterpart in the traditional scheme, or be newly added into the scheme. Section 4 presents the case study of our developed DLVC, with all the design details and experimental results. Section 5 summarizes our perspectives on some open problems for future research, and then concludes this article. Table 1 lists the abbreviations used in this article.

1.3 Preliminaries

In this article, we consider coding methods for natural image/video, i.e., the image/video as-is seen by human taken by daily cameras or mobile phones. Although the methods are usually generally applicable, they have been specifically designed for natural image/video, and may not perform that well for other kinds (e.g. biomedical, remote-sensing).

Currently, almost all the natural image/video is in digital format. A grayscale digital image can be denoted by $\boldsymbol{x} \in \mathbb{D}^{m \times n}$, where m and n are the number of rows (height) and number of columns (width) of the image, and $\mathbb{D}$ is the definition domain of a single picture element (pixel). For example, $\mathbb{D} = \{0, 1, \ldots, 255\}$ is a common setting, where $|\mathbb{D}| = 256 = 2^8$, thus the pixel value

Table 1. List of Abbreviations

Abbreviation	Remark
AVC	Advanced Video Coding, i.e., H.264 [140]
BARC	block adaptive resolution coding
BD-rate	Bjontegaard's delta-rate [13]
BPG	Better Portable Graphics, an image coding format based on HEVC
CNN	convolutional neural network
CTU	coding tree unit
CU	coding unit
DLVC	Deep Learning Video Coding, our developed prototype video codec
GAN	generative adversarial network
HEVC	High Efficiency Video Coding, i.e., H.265 [118]
HM	HEVC reference software
ILF	in-loop filter
JEM	Joint Exploration Model, a video coding software developed by JVET
JPEG^1	Joint Photographic Experts Group, a group of ISO/IEC
JPEG^2	a standard published by ISO/IEC [132]
JVET	Joint Video Experts Team, a team of MPEG and VCEG
LSTM	long short-term memory
MAE	mean-absolute-error
MC	motion compensation
ME	motion estimation
MPEG	Moving Picture Experts Group, a group of ISO/IEC
MSE	mean-squared-error
MS-SSIM	multi-scale SSIM
PSNR	peak signal-to-noise ratio
QP	quantization parameter
ReLU	rectified linear unit [99]
RNN	recurrent neural network
SR	super-resolution
SSIM	structural similarity [137]
VCEG	Video Coding Experts Group, a group of ITU-T
VVC	Versatile Video Coding, an incoming video coding standard

can be represented by an 8-bit integer; accordingly, an uncompressed grayscale digital image has 8 bits-per-pixel (bpp), while compressed bits are definitely less.

A color image is usually decomposed into multiple channels to record the color information. For example, using the RGB color space, a color image can be denoted by $\boldsymbol{x} \in \mathbb{D}^{m \times n \times 3}$, where 3 corresponding to three channels–Red, Green, Blue. Since human vision is more sensitive to luminance than to chrominance, the YCbCr (YUV) color space is much more adopted than RGB, and the U and V channels are typically down-sampled to achieve compression. For example, in the so-called YUV420 color format, a color image can be denoted by $\boldsymbol{X} = \{\boldsymbol{x}_Y \in \mathbb{D}^{m \times n}, \boldsymbol{x}_U \in \mathbb{D}^{\frac{m}{2} \times \frac{n}{2}}, \boldsymbol{x}_V \in \mathbb{D}^{\frac{m}{2} \times \frac{n}{2}}\}$.

A color video is composed by multiple color images, called frames, to record the scene at different timestamps. For example, in the YUV420 color format, a color video can be denoted by $\boldsymbol{V} = \{\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_{T-1}\}$ where T is the number of frames, each $\boldsymbol{X}_i = \{\boldsymbol{x}_Y^{(i)} \in \mathbb{D}^{m \times n}, \boldsymbol{x}_U^{(i)} \in \mathbb{D}^{\frac{m}{2} \times \frac{n}{2}}, \boldsymbol{x}_V^{(i)} \in \mathbb{D}^{\frac{m}{2} \times \frac{n}{2}}\}$. If $m = 1080, n = 1920, |\mathbb{D}| = 2^{10}$, and a video has 50 frames-per-second (fps), then the data rate of the uncompressed video is $1080 \times 1920 \times (10 + \frac{10}{4} + \frac{10}{4}) \times 50 = 1{,}555, 200{,}000$ bits-per-second (bps), about 1.555 Gbps. Obviously, the video should be compressed by a ratio of hundreds to thousands before it can be efficiently transmitted over the current wired and wireless networks.

The existing lossless coding methods can achieve a compression ratio of about 1.5 to 3 for natural image, which is clearly below requirement. Thus, lossy coding is introduced to compress more but at the cost of incurring loss. The loss can be measured by the difference between original and reconstructed images, e.g., using mean-squared-error (MSE) for grayscale image:

$$\mathrm{MSE} = \frac{||\boldsymbol{x} - \boldsymbol{x}_{\mathrm{rec}}||^2}{m \times n} \tag{1}$$

Accordingly, the quality of reconstructed image compared with original image can be measured by peak signal-to-noise ratio (PSNR):

$$\mathrm{PSNR} = 10 \times \log_{10} \frac{(\max(\mathbb{D}))^2}{\mathrm{MSE}} \tag{2}$$

where $\max(\mathbb{D})$ is the maximal value in $\mathbb{D}$, e.g., 255 for 8-bit grayscale image. For color image/video, the PSNR values of Y, U, V are usually separately calculated. For video, the PSNR values of different frames are usually separately calculated and then averaged. There are other quality metrics in replacement of PSNR, such as structural similarity (SSIM) and multi-scale SSIM (MS-SSIM) [137].

To compare different lossless coding schemes, we usually compare the compression ratio, or the resulting rate (bpp, bps, etc.). To compare different lossy coding schemes, it is necessary to take into account not only the rate but also the quality. For example, we can calculate the average rate ratio over a range of quality levels, which is known as Bjontegaard's delta-rate (BD-rate) [13]. There are other important aspects to evaluate image/video coding schemes, including encoding/decoding complexity, scalability, robustness, and so on.

2 REVIEW OF DEEP SCHEMES

In this section, we review some representative coding schemes that are built primarily upon deep networks. Generally speaking, there are two approaches for deep image coding schemes, i.e., pixel probability modeling and auto-encoder. These two approaches are combined together in several deep schemes. In addition, we discuss deep video coding schemes and special-purpose coding schemes, where special-purpose schemes are further categorized into perceptual coding and semantic coding.

2.1 Pixel Probability Modeling

According to Shannon's information theory [112], the optimal method for lossless coding can reach the minimal coding rate $-\log_2 p(x)$ where $p(x)$ is the probability of the symbol x. To reach this target, a number of lossless coding methods have been invented, and arithmetic coding is believed to be among the optimal ones [141]. In essence, given a probability $p(x)$, arithmetic coding ensures the coding rate to be as near as possible to $-\log_2 p(x)$ up to rounding error. Thus, the remaining problem is to find out the probability, which is however very difficult for natural image/video as it has very high dimensions.

One way to estimate $p(\boldsymbol{x})$, where $\boldsymbol{x}$ is an image, is to decompose the image into $m \times n$ pixels and to estimate the probabilities of these pixels one by one (e.g., in the raster scan order). When estimating the probability of one pixel, we take advantage of the previous pixels to predict. This is a typical *predictive coding* strategy. Note that

$$p(\boldsymbol{x}) = p(x_1)p(x_2|x_1)\cdots p(x_i|x_1,\ldots,x_{i-1})\cdots p(x_{m\times n}|x_1,\ldots,x_{m\times n-1}) \tag{3}$$

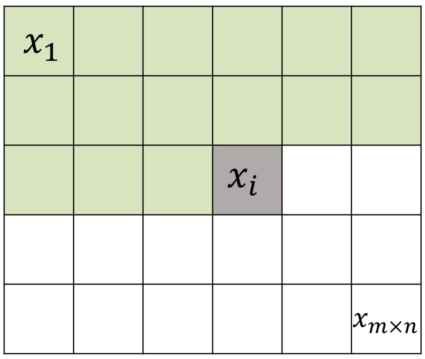


Fig. 1. Illustration of a typical predictive coding scheme, where the pixels are encoded/decoded one by one in the raster scan order. For the pixel x_i (marked gray), all the previous pixels (marked green), i.e., above of x_i and left in the same row of x_i, can be used as condition to predict the pixel value of x_i. The green area is also called the *context* for x_i. For simplification, context can be chosen as a subset of the green area.

which is illustrated in Figure 1. Here the condition for x_i is also called the *context* for x_i. When the image is large, the conditional probability can be difficult to estimate. A simplification is to reduce the range of context, e.g.,

$$p(\boldsymbol{x}) = p(x_1)p(x_2|x_1)\cdots p(x_i|x_{i-k},\ldots,x_{i-1})\cdots p(x_{m\times n}|x_{m\times n-k},\ldots,x_{m\times n-1}) \tag{4}$$

where k is a prechosen constant.

As known, deep learning is good at solving prediction problems, including regression and classification. Therefore, it has been proposed to predict the probability $p(x_i|x_1,\ldots,x_{i-1})$ given the context $x_1,\ldots,x_{i-1}$, using trained deep networks. This strategy is proposed for other kinds of high-dimensional data in as early as 2000 [12], but is applied to image/video until recently. For example, in [62], the probability estimation is considered for binary images, i.e., $x_i \in \{-1,+1\}$, where it suffices to predict a single probability value $p(x_i = +1|x_1,\ldots,x_{i-1})$ for each pixel. This article presents the neural autoregressive distribution estimator (NADE), where a feed-forward network with one hidden layer is used for each pixel, and the parameters are shared across these networks. The parameter sharing also help to speed up the computations for each pixel. A similar work is presented in [40], where the feed-forward network also has connections skipping the hidden layer, and the parameters are also shared. Both [62] and [40] perform experiments on the binarized MNIST dataset.[1] Uria et al. [127] extend the NADE to real-valued NADE (RNADE), where the probability $p(x_i|x_1,\ldots,x_{i-1})$ is made up by a mixture of Gaussians, and the feed-forward network needs to output a set of parameters for the Gaussian mixture model, instead of a single value in NADE. Their feed-forward network has one hidden layer and parameter sharing, but the hidden layer is equipped with rescaling to avoid saturation, and uses rectified linear unit (ReLU) [99] instead of sigmoid. They also consider mixture of Laplacians rather than Gaussians. Experiments are conducted on 8×8 natural image patches, where the pixel value is added with noise and converted to real value. In [128], NADE and RNADE are improved by using different orderings of the pixels as well as using more hidden layers in the network. In [131], RNADE is improved by enhancing the Gaussian mixture model (GMM) with deep GMM.

Designing advanced networks has been an important theme for improving pixel probability modeling. In [120], multi-dimensional long short-term memory (LSTM) based network is proposed, together with mixtures of conditional Gaussian scale mixtures, a generalization of GMM,

[1] http://yann.lecun.com/exdb/mnist/.

for probability modeling. LSTM is a kind of recurrent neural networks (RNNs), and is regarded good at modeling sequential data. The spatial variant of LSTM is used for images. Later, in [129], several different networks are studied, including RNNs and CNNs that are known as PixelRNN and PixelCNN, respectively. For PixelRNN, two variants of LSTM, called row LSTM and diagonal BiLSTM, are proposed, where the latter is specifically designed for images. PixelRNN incorporates residual connections [43] to help train deep networks with up to 12 layers. For PixelCNN, in order to suit for the shape of context (see Figure 1), *masked* convolutions are proposed. PixelCNN is also as deep as having 15 layers. Compared with previous works, PixelRNN and PixelCNN are more dedicated to natural images: they consider pixels as discrete values (e.g., 0, 1, . . . , 255), and predict a multinomial distribution over the discrete values; they deal with color images (in RGB color space); multi-scale PixelRNN is proposed; and they work well on the CIFAR-10 and ImageNet datasets. Quite a number of works follow the approach of PixelRNN and PixelCNN. In [130], Gated PixelCNN is proposed to improve the PixelCNN, and achieves comparable performance with PixelRNN but with much less complexity. In [108], PixelCNN++ is proposed with the following improvements upon PixelCNN: a discretized logistic mixture likelihood is used rather than a 256-way multinomial distribution; down-sampling is used to capture structures at multiple resolutions; additional short-cut connections are introduced to speed up training; dropout is adopted for regularization; RGB is combined for one pixel. In [19], PixelSNAIL is proposed, in which casual convolutions are combined with self attention.

Most of the aforementioned works directly model pixel probability. In addition, pixel probability may be modeled as a conditional one upon explicit or latent representations. That says, we may estimate

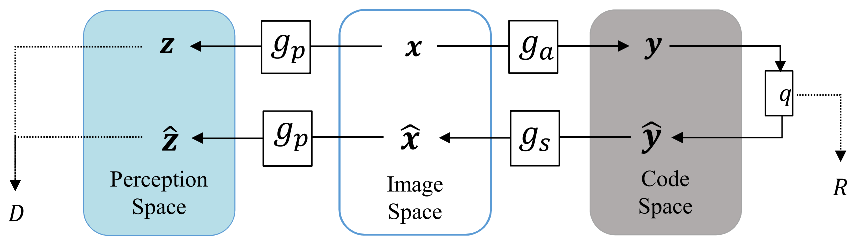
$$p(\boldsymbol{x}|\boldsymbol{h}) = \prod_{i=1}^{m\times n} p(x_i|x_1, \ldots, x_{i-1}, \boldsymbol{h}) \tag{5}$$

where $\boldsymbol{h}$ is the additional condition. Note also that $p(\boldsymbol{x}) = p(\boldsymbol{h})p(\boldsymbol{x}|\boldsymbol{h})$, which means the modeling is split into an unconditional and a conditional. For example, in [130], the additional condition can be image class or high-level image representations that are derived by another deep network. In [60], PixelCNN with latent variables are considered, where the latent variables are derived from the original image: they can be a quantized grayscale version of the original color image, or a multi-resolution image pyramid.

Regarding practical image coding schemes, in [69], a network with *trimmed* convolutions is adopted to predict probabilities for binary data, while a 8-bit grayscale image with the size of $m \times n$ is converted to a binary cube with the size of $m \times n \times 8$ to be processed by the network. The network is similar to PixelCNN but is three dimensional. The trimmed convolutional network-based arithmetic encoding (TCAE) is reportedly better than the previous non-deep lossless coding schemes, such as TIFF, GIF, PNG, JPEG-LS, and JPEG 2000-LS; on the Kodak image set,[2] TCAE achieves a compression ratio of 2.00. In [4], CNN is used in the wavelet transform domain rather than the pixel domain, i.e., CNN is to predict wavelet detail coefficients from coefficients within neighboring subbands.

For video coding, in [55], PixelCNN is generalized to video pixel network (VPN) for the pixel probability modeling of video. VPN is composed of CNN encoders (for previous frames to predict the current frame) and PixelCNN decoders (for prediction inside the current frame). CNN encoders preserve at all layers the spatial resolution of input frames to maximize representational capacity. Dilated convolutions are adopted to enlarge receptive fields and better capture global motion. The outputs of the CNN encoders are combined over time with a convolutional LSTM that also

[2]http://www.r0k.us/graphics/kodak/.



$$\min_{g_a, g_s, q} D(\boldsymbol{z}, \hat{\boldsymbol{z}}) + \lambda R(q(\boldsymbol{y}))$$

Fig. 2. Illustration of a typical transform coding scheme. The original image $\boldsymbol{x}$ is transformed by an *analysis* function g_a to achieve the code $\boldsymbol{y}$. The code $\boldsymbol{y}$ is quantized (denoted by q) and compressed into bits. The number of bits is used to measure the coding rate (R). The quantized code $\hat{\boldsymbol{y}}$ is then inversely transformed by a *synthesis* function g_s to achieve the reconstructed image $\hat{\boldsymbol{x}}$. Both of $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ are further transformed by a same *perceptual* function g_p, resulting in z and $\hat{z}$, respectively. The difference between z and $\hat{z}$ is used to measure the distortion (D).

preserves the resolution. The PixelCNN decoders use masked convolutions and adopt multinomial distributions over discrete pixel values. VPN is experimented on the Moving MNIST and Robotic Pushing datasets.

A distinctive work is reported in [110], where Schiopu et al. investigate a lossless image coding scheme and use CNN to predict pixel value rather than its distribution. The predicted value is subtracted from the true pixel value, resulting in residue that is then coded. In addition, they consider the adaptive selection among the CNN predictor and some non-CNN predictors.

2.2 Auto-Encoder

Auto-encoder originates from the well-known work of Hinton and Salakhutdinov [45], which trains a network for dimensionality reduction and the network consists of the encoding part and the decoding part. The encoding part converts an input high-dimension signal to its low-dimension representation, and the decoding part recovers (not perfectly) the high-dimension signal from the low-dimension representation. Auto-encoder enables automated learning of representations and eliminates the need of hand-crafted features, which is also believed to be one of the most important advantages of deep learning.

It seems quite straightforward to adopt the auto-encoder network for lossy image coding: the encoder and decoder are trained out, and we just need to encode the learned representation. However, the traditional auto-encoder is not optimized for compression, and directly using a trained auto-encoder is not efficient [139]. When we consider the compression requirement, there are several challenges: First, the low-dimension representation shall be quantized then coded, but the quantization step is not differentiable, making it difficult to train the network. Second, lossy coding is to achieve a better tradeoff between rate and quality, so the rate shall be taken into account when training the network, but the rate is not easy to calculate or estimate. Third, a practical image coding scheme needs to consider the variable rate, scalability, encoding/decoding speed, interoperability, and so on. In response to these challenges, a number of works have been conducted especially in recent years.

Indeed, auto-encoder-based image coding schemes follow a typical *transform coding* strategy, where we perform data transformation before coding. A conceptual illustration of auto-encoder-based image coding is shown in Figure 2. The original image $\boldsymbol{x}$ is transformed to $\boldsymbol{y} = g_a(\boldsymbol{x})$, and $\boldsymbol{y}$ is quantized then coded. The decoded $\hat{\boldsymbol{y}}$ is inversely transformed to $\hat{\boldsymbol{x}} = g_s(\hat{\boldsymbol{y}})$. Considering the tradeoff between rate and quality, we can train the network to minimize the joint rate-distortion cost $D + \lambda R$, where D is calculated or estimated as the difference between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ (note that the difference may be calculated or estimated in a perception space), R is calculated or estimated from the quantized code, and λ is the Lagrange multiplier. All of the existing works follow this scheme more or less and differ in their network structure and loss function.

For the network structure, RNNs and CNNs are the widely used two categories. The most representative works of RNN-based methods are as follows. Toderici et al. [122] propose a general framework for variable rate image compression. They use binary quantization to generate codes, and do not consider the rate during training, i.e. the loss is only end-to-end distortion, measured by MSE. Their framework indeed provides a scalable coding functionality, where RNN (specifically LSTM) with convolutional and deconvolutional layers is reported to perform well. They provide results on a large-scale dataset of 32×32 thumbnails. Later, Toderici et al. [123] propose an improved version, where they use a neural network like PixelRNN [129] to compress the binary codes; they also introduce a new gated recurrent unit (GRU) inspired by the residual network (ResNet) [43]. They report better results than JPEG on the Kodak image set using MS-SSIM as quality metric. Johnston et al. [54] further improve the RNN-based method by introducing hidden-state priming into RNN, using an SSIM-weighted loss function, and enabling spatially adaptive bitrates. They achieve better results than BPG on the Kodak image set using MS-SSIM. Covell et al. [24] enable spatially adaptive bitrates by training stop-code tolerant RNNs.

The most representative works of CNN-based methods are as follows: Ballé et al. [9] propose a general framework for rate-distortion optimized image compression. They use multiary quantization to generate integer codes and consider the rate during training, i.e., the loss is the joint rate-distortion cost, where distortion can be MSE or others. To estimate the rate, they use the addition of a random noise to replace the quantization during training, and use the differential entropy of the noisy "code" as a proxy for the rate. As for the network structure, they use the generalized divisive normalization (GDN) transform, which consists of a linear mapping (matrix multiplication) followed by a nonlinear parametric normalization; the effectiveness of the proposed GDN for image coding is verified in [8]. Later, Ballé et al. [10] propose an improved version, where they use three convolutional layers each followed by down-sampling and a GDN operation to implement the transform; accordingly, they use three layers of inverse GDN + up-sampling + convolution to implement the inverse transform. In addition, they design an arithmetic coding method to compress the integer codes. They report better results than JPEG and JPEG 2000 on the Kodak image set using MSE as quality metric. Furthermore, Ballé et al. [11] improve their scheme by incorporating a scale hyper-prior into the auto-encoder, which is inspired by the variational auto-encoder [58]. They use another transform h_a to convert $\boldsymbol{y}$ into $\boldsymbol{w} = h_a(\boldsymbol{y})$, quantize and encode $\boldsymbol{w}$ (transmitted as side information), and use another inverse transform h_s to convert the decoded $\hat{\boldsymbol{w}}$ into the estimated standard deviation of the quantized $\hat{\boldsymbol{y}}$, which is then used during the arithmetic coding of $\hat{\boldsymbol{y}}$. On the Kodak image set and using PSNR as quality metric, their method is only slightly worse than BPG.

Besides [9], several works also concentrate on dealing with the non-differentiable quantization and/or the estimation of rate. Theis et al. [121] adopt a very simple work-around for quantization: quantization is performed as usual in the forward pass, but the gradients are directly passed through the quantization layer in the backward pass. Surprisingly, this work-around works well. In addition, they replace the rate with an upper bound that is differentiable. Dumas et al. [31] consider a stochastic winner-take-all mechanism, where the entries in $\boldsymbol{y}$ with the largest absolute values are kept and the other entries are set to 0; then the entries are uniformly quantized and compressed. Agustsson et al. [2] propose a soft-to-hard vector quantization scheme, where they use a soft quantization (i.e., assigning a representation to multiple codes with different membership values) rather than hard quantization (i.e., assigning a representation to only one code) during training, and they adopt an annealing process to let the soft quantization approach the hard quantization gradually. Note that their scheme takes advantage of vector quantization while other works usually adopt scalar quantization. Li et al. [70] introduce an importance map for rate estimation, where the importance map is quantized to a mask and the mask decides how many bits are kept at each location, thus the sum of the importance map can be used as a rough estimate of the coding rate.

Besides [122], several works also consider the functionality of variable rate with less or no training for different rates. In [121], scale parameters are introduced and a pretrained auto-encoder is fine-tuned for different rates. In [32], a unique learned transform is proposed, together with variable quantization step for different rates. In [15], a multi-scale decomposition transform is trained and optimized for all scales; and rate allocation algorithms are provided to determine the optimal scale of each image block for either a target rate or a target quality factor. Besides, scalable coding is considered in [160] differently from that in [122]. In [160], an image is decomposed into multiple bit-planes, which are transformed and quantized in parallel; bidirectional assembling gated units are proposed to reduce the correlation between different bit-planes.

Several works consider advanced network structures and different loss functions. Theis et al. [121] adopt a sub-pixel structure for computational efficiency. Rippel and Bourdev [106] present a pyramid decomposition followed by inter-scale alignment network, which is lightweight and runs in real-time. They also use a discriminator loss in addition to the reconstruction loss. Snell et al. [114] use the MS-SSIM as loss function instead of MSE or mean-absolute-error (MAE) to train auto-encoders, and they find that MS-SSIM is better calibrated to perceptual quality. Zhou et al. [164] use deeper networks for encoder/decoder and a separate network for post-processing at the decoder side. They also replace the Gaussian model in [11] with the Laplacian model.

Moreover, Cheng et al. [23] apply principle component analysis on the learned representation, which is virtually a second transform. Note that the hyper-prior [11] mentioned before is also a second transform.

2.3 Hybrid Image Coding

Pixel probability modeling represents predictive coding and auto-encoder represents transform coding. These two strategies have their pros and cons. First, the flexibility of predictive coding is higher, because we can select a different predictor for each pixel (e.g., in [110]). However for transform coding, a single transform, though can be quite complicated, is applied to all pixels. Due to the flexibility, predictive coding usually leads to higher compression ratio. Second, predictive coding can be used for both lossless and lossy coding, but transform coding is typically used for lossy coding only, because it is not a trivial task to ensure perfect reconstruction for forward transform and inverse transform. Third, predictive coding implies a sequential encoding/decoding pipeline. For example, when using PixelCNN [129], the pixels shall be decoded one by one, because to decode a pixel we need to know its probability distribution and that probability distribution is predicted from the previously decoded pixel values. On the contrary, when using auto-encoders, all pixels can be decoded at once by using a single inverse transform. That says, transform coding benefits parallel decoding.

Predictive coding and transform coding are not contradictory. Indeed, they can be combined for higher compression efficiency, leading to *hybrid* coding schemes. For example, Mentzer et al. [96] propose a practical lossless image coding scheme, where they use auto-encoders at multiple levels to learn the condition for pixel probability modeling. Mentzer et al. [95] integrate pixel probability modeling (a 3D PixelCNN) into auto-encoder so as to estimate the coding rate and to train the PixelCNN and the auto-encoder jointly. Baig et al. [6] introduce partial-context image inpainting into the variable rate compression framework [122], which is actually to predict a block from the block's context, assuming the blocks are encoded/decoded one by one in the raster scan order (similar to what is shown in Figure 1 but at the block level). The prediction signal is added onto the network output signal to achieve $\hat{\boldsymbol{x}}$, i.e., the transform coding network deals with the prediction residues. Minnen et al. [98] additionally consider rate allocation among the blocks. Klopp et al. [59] also use a spatial predictor, but it is applied to the transformed codes rather than pixels or blocks. Minnen et al. [97] improve upon [11] by augmenting the hyper-prior with the context, i.e., they use not only $\hat{\boldsymbol{w}}$ but also the context to predict the probability of each entry of $\hat{\boldsymbol{y}}$. Lee et al. [64] introduce the context adaptive entropy model into the hyper-prior $\hat{\boldsymbol{w}}$. Minnen et al.'s [97] and Lee's [64] methods outperform BPG on the Kodak image set using PSNR as quality metric, which represent the state of the art by the end of 2018.

2.4 Video Coding

Compared to image coding, video coding calls for efficient methods to remove the redundant content among multiple pictures. For this purpose, inter-picture prediction is a critical issue in video coding. Traditionally, inter-picture prediction is accomplished by block-wise motion estimation and motion compensation [118, 140]. Recently, trained deep networks have been adopted for pixel-wise motion estimation (i.e., optical flow estimation) and motion compensation. Representative works are discussed below.

Chen et al. [18] seems to be the first to report a video coding scheme by using trained deep networks as auto-encoders. Specifically, they divide video frames into 32×32 blocks and for each block they choose one of two modes: intra-coding or inter-coding. If using intra-coding, there is an auto-encoder to compress the block. If using inter-coding, then they perform block-wise motion estimation and compensation using the traditional method, and input the residues to another auto-encoder. For both auto-encoders, the encoded representations are directly quantized and coded by the Huffman method. This scheme is quite rough and does not compete H.264.

Wu et al. [143] propose a video coding scheme with image interpolation, where the key frames (I frames) are first compressed by the deep image coding scheme in [123], and the remaining frames (B frames) are then compressed in a hierarchical order. For each B frame, two compressed frames (either I frames or previously compressed B frames) before and after are used to "interpolate" the current frame: the motion information is used to warp the two compressed frames (i.e., motion compensation), and then the two warped frames are sent as side information to a variable rate image coding scheme that processes the current frame. The scheme is reported to perform on par with H.264.

Chen et al. [22] propose another video coding scheme with the so-called PixelMotionCNN. In their scheme, frames are compressed in the temporal order, and each frame is divided into blocks that are compressed in the raster scan order. Before one frame is compressed, the previous two compressed frames are used to "extrapolate" the current frame. When a block is to be compressed, the extrapolated frame together with the block's context are sent to the PixelMotionCNN to generate a prediction signal for the current block, then the prediction residues are compressed by the variable rate image coding scheme in [123]. This scheme also performs on par with H.264.

Lu et al. [86] propose a real end-to-end deep video coding scheme, which can be viewed as a "deepened" version of the traditional video coding schemes. Specifically in their scheme, for each frame to be compressed, an optical flow estimation module is used to obtain the motion information between the frame and the previous compressed frames. Motion compensation is also performed by a trained network, to generate a prediction signal for the current frame. For the prediction residues and the motion information, two auto-encoders are used to compress them, respectively. The entire network is jointly optimized with a single loss function, i.e., the joint rate-distortion cost. This scheme reportedly achieves better compression efficiency than H.264, and even outperforms HEVC (x265 encoder) when evaluated with MS-SSIM.

Rippel et al. [107] present the to-date most sophisticated deep video coding scheme, which inherits and extends a deepened version of the traditional video coding schemes. Their scheme has the following new features: (1) only one auto-encoder to compress motion information and prediction residues simultaneously; (2) a state that is learned from the previous frames and updated recursively; (3) motion compensation with multiple frames and multiple optical flows; (4) a rate control algorithm. This scheme is reported to outperform HEVC reference software (HM) when evaluated with MS-SSIM.

By the end of 2018, we do not observe any report that a deep video coding scheme can outperform HM when evaluated with PSNR, which seems a hard mission.

2.5 Special-Purpose Coding

Most of the works about deep schemes concern image/video coding for *signal fidelity*, i.e., to minimize the distortion between the original and the reconstructed image/video subject to a given rate, where the distortion can be defined as MSE or other differences. However, if we are not concerned about the fidelity, we may instead be interested in the perceptual naturalness of the reconstructed image/video, or the utility of the reconstructed image/video in semantic analysis tasks. The latter two kinds of quality metrics are termed *perceptual naturalness* and *semantic quality*. There have been a few works that tailor image/video coding for these quality metrics.

2.5.1 Perceptual Coding.

Since the boom of generative adversarial network (GAN) [37], deep networks are known to be capable in generating perceptually natural images. Leveraging this capability at the decoder side can surely improve the perceptual quality of decoded images. Unlike the generator in normal GANs, the decoder should also ensure the decoded images to be similar to original images, which raises a problem of controlled generation and the encoder actually provides the control signal in the coded bits.

Inspired by the variational auto-encoder (VAE) [58], Gregor et al. [39] propose Deep Recurrent Attentive Writer (DRAW) for image generation, which extends the traditional VAE by using RNNs as encoder and decoder. Unfolding the encoder RNN produces a series of latent representations. Then, Gregor et al. [38] introduce convolutional DRAW, and observe that it is able to transform an image into a series of increasingly detailed representations, ranging from global conceptual aspects to low-level details. Thus, they suggest a conceptual compression scheme, whose one benefit is to achieve plausible reconstruction images at very low bit rates.

It has been realized that perceptual naturalness can be evaluated by the discriminator in GAN [14]. Several works are devoted to deep coding schemes for perceptual quality using the discriminator loss solely or jointly with MSE or other losses. For example, Santurkar et al. [109] propose the so-called generative compression schemes for both image and video. For image, they first train a canonical GAN, then they use the generator as the decoder, fix it, and train the encoder to minimize a sum of MSE and feature loss. For video, they reuse the encoder and decoder trained for image, transmit only a few frames, and restore the other frames at the decoder side via interpolation. Their schemes are able to achieve very high compression ratio. Kim et al. [57] build a new video compression scheme, where a few key frames are normally compressed (by H.264) and the other frames are extremely compressed. Indeed, edges are extracted from the down-sampled non-key frames and transmitted. At the decoder side, the key frames are first reconstructed, then edges are similarly extracted from them. A conditional GAN is trained with the reconstructed key frames where edge is the condition. Then the conditional GAN is used to generate the non-key frames. Again, their scheme performs well at very low bit rates.

2.5.2 Semantic Coding. A few works have been conducted on deep coding schemes that preserve the semantic information or that concern the semantic quality.

Agustsson et al. [3] present a GAN-based image compression scheme for extremely low bit rates. The scheme combines auto-encoder and GAN, collapsing the decoder and the generator into one. In addition, a semantic label map can be used as an additional input to the encoder, and as a condition for the discriminator. It is reported that the proposed scheme reconstructs images with higher semantic quality, in the sense that the semantic segmentation on these images is more accurate than that on BPG-compressed images at the same rate.

Luo et al. [87] propose a concept of deep semantic image compression (DeepSIC), which incorporates the semantic information (e.g., classes) into the coded bits. There are two versions of DeepSIC, both based on auto-encoder. In the one version, the semantic information is extracted from the representation $\boldsymbol{y}$, and encoded into the bits. In the other version, the semantic information is not encoded, but extracted at the decoder side from the quantized representation $\hat{\boldsymbol{y}}$. Torfason et al. [124] investigate performing semantic analysis tasks (classification and semantic segmentation) from the quantized representation rather than from the reconstructed image. That means, the decoding process is omitted. They show that the classification and segmentation accuracy values are very close between the representation and the image, but the computational complexity is reduced significantly. Zhang et al. [156] study a deep image coding scheme for simultaneous compression and retrieval. Their motivation is that the coded bits can be used not only for reconstructing image but also for retrieving similar images *without decoding*. They use an auto-encoder to compress image into bits, and use a revised classification network to extract binary features. Then they combine the two parts of bits, and fine-tune the feature extraction network for image retrieval. Their results indicate that at the same rate, the reconstructed images are better than JPEG-compressed ones, and the retrieval accuracy improves due to the fine-tuning.

Akbari et al. [5] design a scalable image coding scheme where the coded bits consist of three layers. The first layer is the semantic segmentation map coded losslessly. The second layer is a down-sampled version of the original image also coded losslessly. With the first two layers, a network is trained to predict the original image and the prediction residues are coded by BPG as the third layer. This scheme is reported to outperform BPG on the Kodak image set when evaluated with PSNR and MS-SSIM.

Chen and He [21] consider deep coding for facial images with semantic quality metric instead of PSNR or perceptual quality. For this purpose, their loss function has three parts: MAE, discriminator loss, and a semantic loss, where the semantic loss is to project the original and reconstructed images into a compact Euclidean space through a learned transformation, and to calculate the Euclidean distance between them. Accordingly, their scheme performs very well when evaluated with face verification accuracy at the same rate.

2.6 Remarks

To summarize, the existing deep image/video coding schemes can be categorized into predictive, transform, and hybrid coding schemes. Hybrid schemes that combine both predictive and transform coding are more appealing to pursue higher compression efficiency, especially for video coding. There can be different ways to combine. One way is first to predict and then to transform the prediction residues. Another way is first to transform and then to predict the probabilities of the transformed codes one by one. Generally speaking, the method of combination largely decides the network structure of deep schemes. Searching for a better network structure may be a long-standing topic for deep schemes.

From another point of view, deep image/video coding schemes can be optimized for signal fidelity, perceptual naturalness, semantic quality, or their combinations. Currently, we observe that

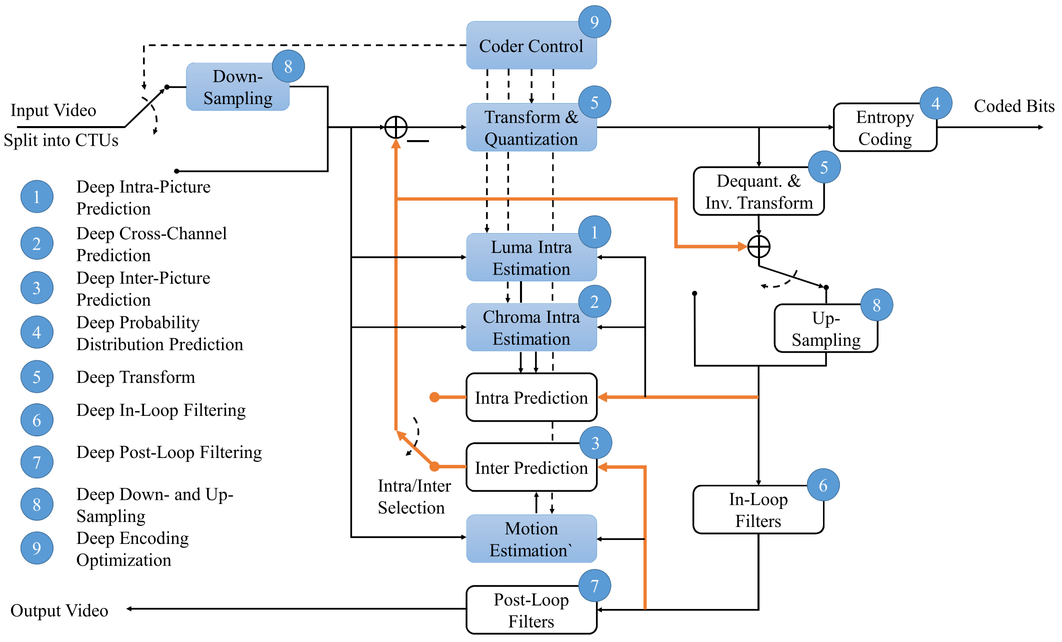


Fig. 3. Illustration of a traditional hybrid video coding scheme as well as the locations of deep tools inside the scheme. Note that the yellow lines indicate the flow of prediction, and the blue boxes indicate the tools that are used at the encoder side only.

for different optimization targets, the loss function has been customized during training. In addition, maybe the network structure is also customized for different targets.

3 REVIEW OF DEEP TOOLS

In this section, we review some representative works about using trained deep networks as tools within the traditional coding schemes or together with traditional coding tools. Generally speaking, the traditional video coding schemes adopt a hybrid coding strategy, i.e., a combination of predictive coding and transform coding. As depicted in Figure 3, an input video sequence is divided into pictures, pictures are divided into blocks (the largest block is called CTU, which can be divided into smaller CUs, in HEVC [118]), and blocks are divided into channels (i.e., Y, U, V). The pictures/blocks/channels are compressed in a predefined order, and the previously compressed ones can be used to predict the following ones, which is known as intra-picture prediction (between blocks), cross-channel prediction (between channels), and inter-picture prediction (between pictures), respectively. The prediction residues are then transformed and quantized and entropy coded to achieve the final bits. Some auxiliary information such as block partition and prediction mode is also entropy coded into the bits (not shown in the figure). Probability distribution prediction is used in the entropy coding step. Since the quantization step loses information and may cause artifacts, filtering is proposed to enhance the reconstructed video, which may be performed in-loop (before predicting the next picture) or out-of-loop (before output). In addition, to reduce the data volume, the pictures/blocks/channels may be down-sampled before being compressed, and up-sampled afterwards. Finally, the encoder needs to control the different modules and combine them to achieve a tradeoff between coding rate, quality, and computational speed. Encoding optimization is an important theme in practical coding systems.

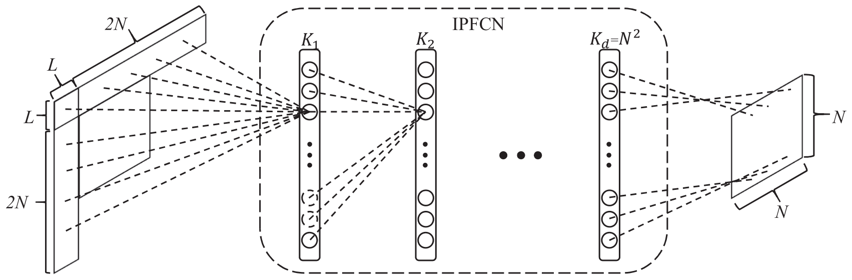


Fig. 4. Illustration of a fully connected network for intra prediction (IPFCN).

Trained deep networks can act as almost all of the modules shown in Figure 3, where we have indicated different locations for deep tools. In the following, we will review the works about deep tools according to where they are used in the entire scheme.

3.1 Intra-Picture Prediction

Intra-picture prediction, or intra prediction for short, is a tool to predict between blocks inside the same picture. H.264 introduces intra prediction with several predefined prediction modes, such as DC prediction and extrapolation along different directions [140]. The encoder can choose a prediction mode for each block and signal the choice to the decoder. To decide mode, it is a common strategy to compare the coding rate and distortion of different modes and to select the mode with the minimal rate-distortion cost. In HEVC, more prediction modes are introduced [118].

Li et al. [67] propose a fully connected network for intra prediction that is depicted in Figure 4. For the current $N \times N$ block, they use L rows above and L columns to the left, in total $4NL + L^2$ pixels as context. They use an image set known as the New York City Library to generate training data, in which the raw image is compressed at different quantization parameters (QPs). When training the network, they investigate two strategies: the first is to train a single model with all training data, and the second is to split the training data into two groups by considering the HEVC prediction modes, and to train two models, respectively. The strategy of two models turns out better for compression. They integrate the trained networks as new prediction modes along with the HEVC modes. They report around 3% BD-rate reduction than HM.

Pfaff et al. [103] also adopt fully connected network for intra prediction, but propose to train multiple networks as different prediction modes. In addition, they propose to train a separate network whose input is also the block's context but output is the predicted likelihood of different modes. Moreover, they propose to use a different transform for each of the network-based prediction modes. Their reported performance is high: around 6% BD-rate reduction than an improved version of HM (with advanced block partitioning).

Hu et al. [47] devise a progressive spatial RNN for intra prediction. Different from the above works, they propose to leverage the sequential modeling capacity of RNN to generate prediction progressively from the context to the block. In addition, they suggest the use of sum-of-absolute-transformed-difference (SATD) as the loss function and argue that SATD correlates better to the rate-distortion cost.

Cui et al. [25] consider a CNN for intra prediction, or more specifically, intra prediction refinement. They use the HEVC prediction modes to generate prediction, and then use a trained CNN to refine the prediction. Note that the CNN has not only the HEVC prediction but also the context as its input. This method seems to achieve only marginal gain.

3.2 Inter-Picture Prediction

Inter-picture prediction, or inter prediction for short, is a tool to predict between video frames so as to remove the redundancy along the temporal dimension. Inter prediction is the kernel of video coding and it largely decides the compression efficiency of a video coding scheme. In the traditional video coding schemes, inter prediction is mostly fulfilled by block-level motion estimation (ME) and motion compensation (MC). Given a reference frame and a block to be coded, ME finds the location in the reference frame where the content is the most similar to that inside the to-be-coded block and MC retrieves the content at the found location so as to predict the block. Many techniques have been proposed to improve block-level ME and MC, such as using multiple reference frames, bi-directional inter prediction (i.e., using two reference frames jointly), fractional-pixel ME and MC, and so on.

Inspired by the multiple reference frames, Lin et al. [76] propose a new inter prediction mechanism by extrapolating the multiple reference frames. Specifically, they adopt a Laplacian pyramid of GANs to extrapolate a frame from the previously compressed four frames. This extrapolated frame serves as another reference frame. They report around 2% BD-rate reduction than HM.

Inspired by the bi-directional inter prediction, Zhao et al. [163] propose a method to enhance the prediction quality. The previous bi-directional prediction simply computes a linear combination of two prediction blocks. They propose to employ trained CNN to combine the two prediction blocks in a nonlinear and data-driven manner. A similar work is reported in [162].

Inspired by the fractional-pixel ME and MC, a number of works are conducted on the fractional-pixel interpolation problem, which aims at generating imaginary pixels at fractional locations on the reference frame because the motion between two frames is not aligned to integer pixels. Here, a major difficulty is how to prepare training data because fractional pixels are imaginary. Yan et al. [150] propose to use a CNN for half-pixel interpolation, where they suggest a method that blurs a high-resolution image and then samples pixels from the blurred image: odd locations as integer pixels and even locations as half pixels. This method is used in [82], where the authors analyze the effect of different blurring degrees. Zhang et al. [154] propose another method, which formulates the fractional interpolation as a resolution enhancement problem. Thus, they down-sample high-resolution images to achieve training data. Yan et al. [148] consider a different formulation, treating the fractional-pixel MC as an inter-picture regression problem. They use video sequences to retrieve training data, where they rely on the fractional-pixel ME to align different frames, and use reference frame as integer pixels and current frame as fractional pixels. Yan et al. [147, 149] further discover a key characteristic of the fractional interpolation problem, namely its invertibility: if fractional pixels can be interpolated from integer pixels, then integer pixels should also be interpolated from fractional pixels. Based on the invertibility, they propose an unsupervised manner to train CNN for fractional-pixel interpolation.

In addition to the improvements of inter prediction methods, another approach is considered where intra and inter predictions are combined. Specifically, the generation of the prediction signal is based on not only the reference frame but also the context in the current frame. For example, Huo et al. [48] propose to use a trained CNN to refine the inter prediction signal. They find that using the context of the to-be-predicted block can improve the prediction quality. Similarly, Wang et al. [135] also refine the inter prediction signal by a CNN, where the CNN inputs include the inter prediction signal, the context of the current block, and the context of the inter prediction block.

3.3 Cross-Channel Prediction

Cross-channel prediction is to predict between different channels. In the YUV format, the luma channel (Y) is usually coded before the chroma channels (U and V). Thus, it is possible to predict

U from Y, and to predict V from Y and U. A traditional method, known as Linear Model (LM), is intended for cross-channel prediction. The key idea of LM is that chroma can be predicted from luma using a linear function, but the coefficients of the function are not transmitted; instead, they are estimated from the context by performing a linear regression. The linear assumption seems over-simplified.

Baig and Torresani [7] investigate colorization for image compression. Colorization is to predict chroma from luma, which is an ill-posed problem because one luma value can correspond to multiple chroma values. Accordingly, they propose a tree-structured CNN, which is able to generate multiple predictions (called *multiple hypotheses*) given one grayscale image as input. When used for compression, the trained CNN is applied at the encoder side, and the branch that produces the best prediction signal is encoded as side information for decoder. They integrate the method into JPEG, without changing the coding of luma, and experimental results show that the proposed method outperforms JPEG for chroma coding.

Li et al. [72] propose a cross-channel prediction method analogous to LM. In particular, they design a hybrid neural network consisting of a fully connected part and a convolutional part. The former is used to process the context, including three channels, and the latter is to process the luma channel of the current block. Twofold features are fused to get the final prediction. This method outperforms LM by providing more than 2% BD-rate reduction for chroma coding.

3.4 Probability Distribution Prediction

As mentioned before, accurate probability estimation is the key problem in entropy coding. Thus, several works have been done to utilize deep learning for probability distribution prediction to improve the entropy coding efficiency. These works deal with different parts of the information. For example, the intra prediction mode of each block is required to be sent to decoder, and Song et al. [116] design a CNN to predict the probability distribution of the intra prediction mode based on the context. Similarly, Pfaff et al. [103] predict the probability distribution of the intra prediction mode based on the context, but using a fully connected network. If an encoding/decoding scheme allows multiple transforms and each block can be assigned a transform mode, then Puri et al. [105] propose to use a CNN to predict the probability distribution of the transform mode, which is based on the quantized transform coefficients. Ma et al. [89] consider the entropy coding of the quantized transform coefficients, especially the DC coefficients, in the HEVC framework. They design a CNN to predict the probability distribution of the DC coefficient of a block, from the context of the block as well as the AC coefficients of the block. They also extend the work to handle AC coefficients in a similar manner [88], and to handle elements of inter prediction information such as motion vectors [90]. Another work was reported in [91], which deals with the wavelet transform coefficients in the JPEG2000 framework.

3.5 Transform

Transform is an important tool in the hybrid video coding framework to convert signal (usually residues) into coefficients that are then quantized and coded. At the very beginning, video coding schemes adopt discrete cosine transform (DCT), which is then replaced by integer cosine transform (ICT) in H.264. HEVC also adopts ICT but additionally uses integer sine transform for 4×4 luma blocks. Adaptive multiple transforms and secondary transform are also studied. Nonetheless, all these transforms are still very simple.

Inspired by auto-encoder, Liu et al. [79] propose a CNN-based method to achieve a DCT-like transform for image coding. The proposed transform consists of a CNN and a fully connected layer, where the CNN is to preprocess the input block and the fully connected layer is to fulfill the *transform*. In their implementation, the fully connected layer is initialized by the transform matrix of DCT, but then is trained together with the CNN. They use a joint rate-distortion cost to train the network, where rate is estimated by the l_1-norm of the quantized coefficients. They also investigate asymmetric auto-encoders, i.e., the encoding part and decoding part are not symmetric, different from the traditional auto-encoders. Their experimental results show that the trained transform is better than the fixed DCT, and the asymmetric auto-encoders can be useful to achieve a tradeoff between compression efficiency and encoding/decoding time.

3.6 Post- or In-Loop Filtering

Most of the widely used image and video coding schemes are lossy coding ones, i.e., the reconstructed image/video is not exactly the original image/video, for the sake of compression. The loss is usually due to the quantization process shown in Figure 3. When the quantization step is large, the loss is large too, which may lead to visible artifacts in the reconstructed image/video, such as blocking, blurring, ringing, color shift, and flickering. Filtering is the tool to reduce these artifacts, to improve the quality of the reconstructed image/video, and thus to improve the compression efficiency indirectly. For image, the filtering is also known as post-processing because it does not change the encoding process. For video, the filtering is divided into in-loop and out-of-loop, depending on whether the filtered frame is used as reference for the following frames. In HEVC, two in-loop filters are presented, namely deblocking filter (DF) [100] and sample-adaptive offset (SAO) [34].

Post- or in-loop filtering occupies the majority of the related works about deep learning-based image/video coding. Earlier works have focused on post-filtering for image coding, especially JPEG. For example, Dong et al. [28] propose a 4-layer CNN for compression artifacts reduction, namely ARCNN. ARCNN achieves more than 1dB improvement in PSNR than JPEG on the five classical test images when the quality factor (QF) is between 10 and 40. Li et al. [68] use a 20-layer CNN that achieves better performance than ARCNN. Cavigelli et al. [16] use a 12-layer CNN with hierarchical skip connections and test for higher QF from 40 to 76. Wang et al. [138] leverage the prior knowledge of JPEG compression, i.e., quantization of the DCT coefficients of 8×8 blocks, and propose a dual-domain (pixel domain and transform domain) based method. They achieve both higher quality and less computing time than ARCNN. Dual-domain processing is also studied in [17], [41], and [157]. Guo and Chao [42] propose a one-to-many network, which is trained by a combination of perceptual loss, naturalness loss, and JPEG loss. Another work about loss function is presented in [35], which suggests the usage of discriminator loss like in GAN. Ororbia et al. [101] propose an iterative post-filtering method by using a trained RNN. Recently, several works also treat JPEG post-filtering as an image restoration task, like denoising or super-resolution, and propose one network for multiple tasks [20, 84, 119, 153, 155, 159]. Ma et al. [91] present a post-filtering method for JPEG2000, which also achieves significant quality improvement.

Later on, works are more and more conducted for out-of-loop filtering in video coding, especially HEVC. Dai et al. [26] propose a 4-layer CNN for post-filtering of intra frames, where the CNN has variable filter size and residue connection, and is called VRCNN. Wang et al. [133] use a 10-layer CNN for out-of-loop filtering, where they train a CNN to filter one image and used the trained CNN on the video frames individually. Yang et al. [151] propose to train different CNN models for I frames and P frames respectively, and verify the benefit. Jin et al. [53] suggest the use of a discriminator loss in addition to the MSE loss. Li et al. [66] propose to transmit some side information to decoder to select one model for each frame from a previously trained set of models. In addition, Yang et al. [152] propose to utilize the inter-picture correlation during the post-filtering process by inputting multiple neighboring frames into the CNN to enhance one frame. Wang et al. [134] also consider the inter-picture correlation, but using a multi-scale convolutional LSTM. Although the aforementioned works take only the decoded frames as input to the CNN, He et al.

[44] propose to input the block partition information together with decoded frame into the CNN, Kang et al. [56] also input the block partition information into the CNN and design a multi-scale network, Ma et al. [92] input the intra prediction signal and the decoded residual signal into the CNN, and Song et al. [117] input the QP plus the decoded frame into the CNN (they also quantize the network parameters to ensure consistency between different computing platforms). A different work is presented in [125], which does not enhance the decoded frames directly; instead, they propose to calculate the compression residues (i.e., the original video minus the decoded video, to be distinguished from prediction residues) at the encoder side, and train an auto-encoder to encode the compression residues and send to the decoder side. Their method is reported to perform well on domain-specific video sequences, e.g., in video game streaming services.

It is more challenging to integrate CNN-based filter into the coding loop because the filtered frame will serve as reference and will affect the other coding tools. Park and Kim [102] train a 3-layer CNN as an in-loop filter for HEVC. They train two models for two QP ranges: 20–29 and 30–39, respectively, and use one model for each frame according to its QP. The CNN is applied after DF, and SAO is turned off. They also design two cases to apply the CNN-based filter: in the one case, the filter is applied on specified frames based on picture order count (POC); in the other, the filter is tested for each frame and if it improves quality then it is applied, one binary flag for each frame is signaled to decoder in this case. Meng et al. [94] use an LSTM as an in-loop filter, which is applied after DF and before SAO in HEVC. The network has decoded frame together with block partition information as its input, and is trained with a combination of MS-SSIM loss and MAE loss. Zhang et al. [158] propose a residual highway CNN (RHCNN) for in-loop filtering in HEVC. RHCNN-based filter is applied after SAO. They train different RHCNN models for I, P, and B frames, respectively. They also divide QPs into several ranges and train a separate model for each range. Dai et al. [27] propose a deep CNN called VRCNN-ext for in-loop filtering in HEVC. They design different strategies for I frames and P/B frames: CNN-based filter replaces DF and SAO for I frames, but is applied after DF and before SAO for P/B frames with CTU- and CU-level control. At CTU-level, one binary flag for each CTU is signaled to control the on/off of CNN-based filter; if the flag is off, then at CU-level, a binary classifier is used to decide whether to turn on CNN-based filter for each CU. Wang et al. [136] propose a dense residual network (DRN) that consists of multiple dense residual units for in-loop filtering in HEVC. They also train different models for different QPs. Jia et al. [49] also consider a deep CNN for in-loop filtering in HEVC. The filter is applied after SAO and controlled by frame- and CTU-level flags. If frame-level flag is "off," then the corresponding CTU-level flags are omitted. In addition, they train multiple CNN models and train a content analysis network that decides one model for each CTU, which saves the bits of CNN model selection.

3.7 Down- and Up-Sampling

A trend of the video technology is to increase the resolution at different dimensions, such as spatial resolution (i.e., number of pixels), temporal resolution (i.e., frame rate), and pixel value resolution (i.e., bit-depth). The increasing resolution results in multiplied data volume, which raises a great challenge to video transmission systems. When the bandwidth for transmission is limited (e.g., using 2G or 3G mobile network), a common practice is to decrease video resolution before encoding and to increase video resolution back after decoding. This is known as the down- and up-sampling-based coding strategy. The down- and up-sampling can be performed in the spatial domain, the temporal domain, the pixel value domain, or a combination of these domains. Traditionally, the down- and up-sampling filters are often handcrafted. Recently, it is proposed to train deep networks as down- and up-sampling filters for efficient video coding. There are two categories of related works.

The first category is focused on training deep networks as up-sampling filters only, while still using handcrafted down-sampling filters. This is inspired by the success of super-resolution, e.g., [29]. For example, in [33], a dual-network-based up-sampling is proposed. There are two networks, one for compression artifacts reduction, and the other for up-sampling, and the two networks are optimized stepwise. In [1], a joint spatial and pixel value down-sampling is proposed, where the spatial down-sampling is achieved by a handcrafted low-pass filter and the pixel value down-sampling is achieved by bitwise right shift. At the encoder side, a support vector machine is used to decide whether to perform down-sampling for each frame. At the decoder side, a CNN is trained to up-sample the decoded video to its original resolution. In [74], Li et al. only consider spatial down-sampling which is also performed by a handcrafted filter, and train a CNN for up-sampling. But unlike [1], they propose a block adaptive resolution coding (BARC) framework. Specifically, for each block inside a frame, they consider two coding modes: down-sampling then coding and direct coding. The encoder can choose a mode for each block and signal the chosen mode to the decoder. In addition, in the down-sampling coding mode, they further design two sub-modes: using a handcrafted simple filter for up-sampling, and using the trained CNN for up-sampling. The sub-mode is also signaled to the decoder. Li et al. [74] investigate BARC only for I frames. Later, Lin et al. [77] extend the BARC framework for P and B frames and build a complete BARC-based video coding scheme. While the aforementioned works perform down-sampling in the pixel domain, Liu et al. [83] propose down-sampling in the residue domain, i.e., they down-sample the inter prediction residues, and they up-sample the residues by a trained CNN with considering the prediction signal. They also follow the BARC framework.

The second category trains not only up-sampling but also down-sampling filters to allow for more flexibility. For example, in [50], a compression framework with two CNNs is studied. The first CNN down-samples an image, the down-sampled image is then compressed by an existing image encoder (such as JPEG and BPG), and then decoded, the second CNN up-samples the decoded image. One drawback of this framework is that it cannot be trained end-to-end because the image encoder/decoder is not differentiable. To address this problem, Jiang et al. [50] decide to optimize the two CNNs alternatively. Zhao et al. [161] use a virtual codec that is actually a CNN to approximate the functionality of–and thus replace–the encoder/decoder; they also insert a CNN to perform post-processing before the up-sampling CNN; their scheme is fully convolutional and can be trained end-to-end. Moreover, Li et al. [73] simply remove the encoder/decoder and keep only the two CNNs during training; considering that the down-sampled image will be compressed, they propose a novel regularization loss for training, which requires the down-sampled image to be not quite different from the ideal low-pass and decimated (which is approximated by a handcrafted filter) image. The regularization loss is verified to be useful when training down- and up-sampling CNNs jointly for image coding.

3.8 Encoding Optimizations

The aforementioned deep tools are intended for increasing the compression efficiency, especially for reducing bitrate while keeping the same PSNR. There are some other deep tools that target different aspects. In this subsection, we review several deep tools for three different objectives: fast encoding, rate control, and region-of-interest (ROI) coding. Since these tools are used only at the encoder side, we call them encoding optimization tools in summary.

3.8.1 Fast Encoding. Regarding the state-of-the-art video coding standards, H.264 and HEVC, the decoder is computationally simple, but the encoder is much more complex. This is because more and more coding modes are introduced into the video coding standards, and each block can be assigned a different mode. The mode of each block is signaled to the decoder, so the decoder only needs to compute the given mode. But to find the mode for each block, the encoder usually needs to compare the multiple optional modes and select the optimal one, where optimality is claimed in the rate-distortion sense. Therefore, if the encoder performs an exhaustive search, then the compression efficiency is the highest, but the computational complexity may be also very high. Any practical encoder will adopt heuristic algorithms to search for a better mode, where machine learning, especially deep learning, can help.

Liu et al. [85] present a hardware design for HEVC intra encoder, where they adopt a trained CNN to help decide CU partition mode. Specifically, in HEVC intra coding, a CTU is split into CUs recursively to form a quadtree structure. Their trained CNN will decide whether to split a 32×32/16×16/8×8 CU or not based on the content inside the CU and the specified QP. Actually, this is a binary decision problem. Xu et al. [146] additionally consider HEVC inter encoder, and propose an early-terminated hierarchical CNN and an early-terminated hierarchical LSTM to help decide CU partition mode, for I frames and P/B frames, respectively. Jin et al. [52] also consider the CU partition mode decision but for the incoming VVC rather than HEVC, because in VVC a quadruple-tree-binary-tree (QTBT) structure is designed for CU partition, which is more complex than that in HEVC. They train a CNN to perform 5-way classification for a 32×32 CU, where different classes indicate different tree depths. Xu et al. [145] investigate the CU partition mode decision for H.264 to HEVC transcoding. They design a hierarchical LSTM network to predict the CU partition mode from the features extracted from H.264 coded bits.

Song et al. [115] study a CNN-based method for fast intra prediction mode decision in HEVC intra encoder. They train a CNN to derive a list of most probable modes for each 8×8/4×4 block based on the content and the specified QP, and then choose a mode from the list by the normal rate-distortion optimized process.

3.8.2 Rate Control. Given a limited transmission bandwidth, video encoder tries to produce bits that do not overflow the bandwidth. This is known as the *rate control requirement.*

One traditional rate control method is to allocate bits to different blocks according to the R-λ model [65]. In that model, each block has two parameters α and β that are to be determined. Previously, the parameters are estimated by an empirical formula. In [71], Li et al. propose to train a CNN to predict the parameters for each CTU. Experimental results show that the proposed method achieves higher compression efficiency as well as lower rate control error.

Hu et al. [46] attempt to leverage reinforcement learning methods for intra-frame rate control. They draw an analogy between the rate control problem and the reinforcement learning problem: the texture complexity of the blocks and bit balance are regarded as the environment state, the quantization parameter is regarded as an action that an agent needs to take, and the negative distortion of the blocks is regarded as an immediate reward. They train a neural network as the agent.

3.8.3 ROI Coding. ROI refers to the regions of interest in an image. In image compression, it is often required that the content in ROI shall be of high quality and the content not in ROI can be of low quality. Many image coding schemes, such as JPEG and JPEG 2000, support ROI coding. Then, how to identify the ROI is a research problem and has been addressed by using deep learning. Prakash et al. [104] propose a CNN-based method to generate a multi-scale ROI (MS-ROI) map to guide the following JPEG coding. They use a trained image classification network on an image, pick the top five classes predicted by the network, and identify the regions that correspond to these classes. Thus, their MS-ROI map indicates salient regions that are related to semantic analysis.

3.9 Remarks

The existing deep tools can be categorized into three groups. The first group is to increase compression efficiency usually at the cost of increased encoding/decoding time. This group consists of various prediction tools, transform tools, filtering tools, and down- and up-sampling tools. According to the reported works, the most promising tools, i.e., the tools that may improve compression performance significantly, seem to be post- or in-loop filtering and probability distribution prediction. Down- and up-sampling is also very promising for low bit rate coding. The second group is for improving encoding speed usually at the cost of decreased compression efficiency. Such tools are very probable to replace their rivals because they indeed achieve better tradeoff between encoding time and compression performance. The third group is for functionalities such as rate control and ROI. They are also promising, but need to be further demonstrated.

From the deep learning or machine learning point of view, different deep tools belong to either one of two paradigms: supervised learning and unsupervised learning. In supervised learning, a deep network is trained with paired data, and the network is expected to learn a mapping from input to output. Almost all of the prediction tools, filtering tools, and fast encoding tools belong to supervised learning. In unsupervised learning, a deep network is trained with "unlabeled" data, and the network is expected to identify the inherent characteristics of the data. Auto-encoding is a good example of unsupervised learning. Accordingly, transform tools and down-sampling tools belong to unsupervised learning. Comparing the two paradigms, deep networks for supervised learning have been investigated much more than for unsupervised learning. Thus, we anticipate that unsupervised deep tools, especially transform tools, may be developed substantially in the near future.

4 CASE STUDY OF DLVC

We now turn to the case study of our developed DLVC, a prototype video codec. Indeed, DLVC was developed as a proposal in response to the joint call for proposals (JCfP) on video compression with capability beyond HEVC [111]. Now the source code of DLVC has been released for future research.[3] DLVC is crafted upon the JEM software, contains a number of improvements than JEM, and especially has two deep coding tools: CNN-based in-loop filter (CNN-ILF) and CNN-based block adaptive resolution coding (CNN-BARC), both of which are based on trained CNN models. The scheme of DLVC is illustrated in Figure 5. In this section, we focus on the two deep tools. More technical details about DLVC can be found in the technical report [144].

4.1 CNN-Based In-Loop Filter

As mentioned in Section 3.6, a great number of works have been conducted on using trained CNN models for post- or in-loop filtering. CNN-ILF represents our efforts at this aspect.

The network structure of our proposed CNN-ILF is illustrated in Figure 6. Inspired by the SR network in [75], we design a deep CNN having 16 residual blocks (ResBlocks) and 2 convolutional layers, in total 34 layers. Each ResBlock consists of 2 convolutional layers separated by a ReLU mapping, and a skip connection. The entire network has a global skip connection from the input to the output. These skip connections are crucial to train an efficient network and to accelerate the convergence in training.

To train the network, we have used a set of natural images and compressed each image by the DLVC intra coding (turning off all in-loop filters) at different QPs. For each QP, we train a separate model. We only use the luma channel for training but the trained models are used for both luma and chroma channels during compression. We divide images into 70×70 sub-images and

[3] https://github.com/fvc2018/dlvc, http://dlvc.bitahub.com/.

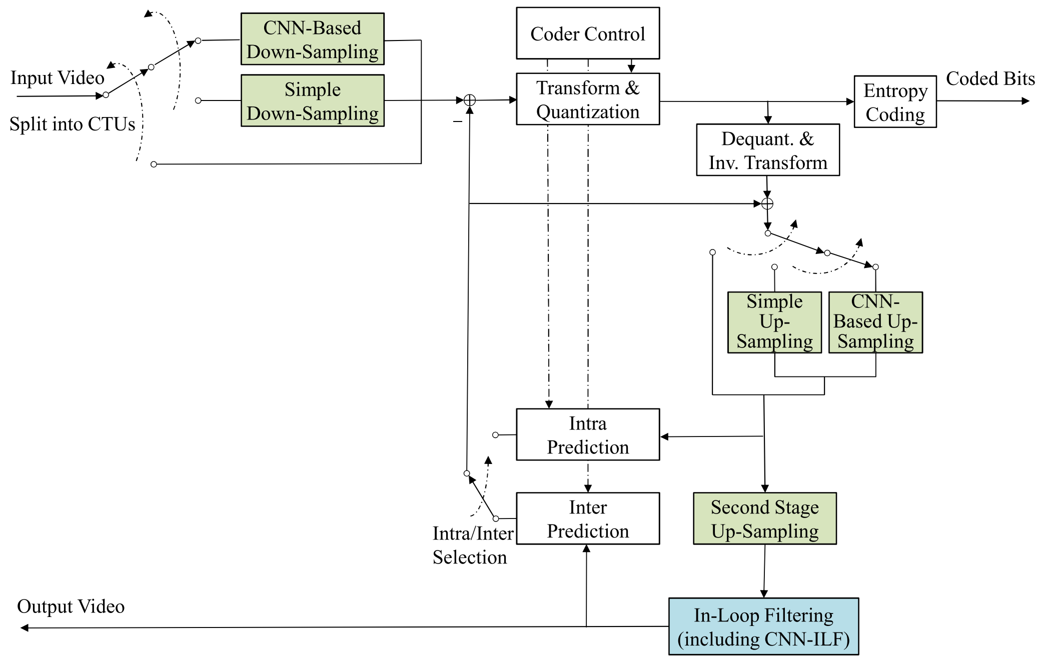


Fig. 5. Illustration of the developed DLVC scheme. Inside the blue block is the proposed CNN-ILF. The green blocks correspond to CNN-BARC.

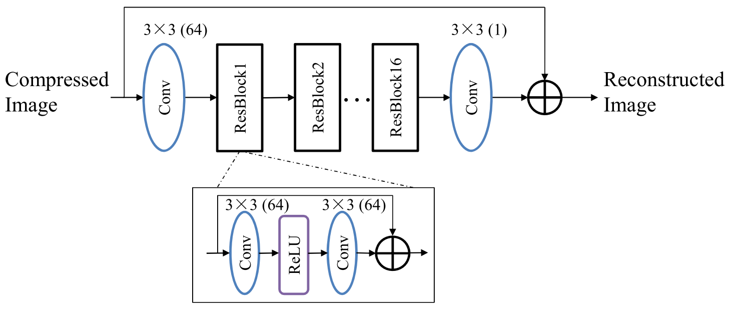


Fig. 6. The network structure of the proposed CNN-ILF. The numbers shown above each convolutional (Conv) layer indicate the kernel size (e.g., 3×3) and the number of output channels (e.g., 64). The symbol ⊕ denotes element-wise sum.

shuffle the sub-images to prepare training data. The loss function is MSE, i.e., the error between the network-output image and the original uncompressed image. We use the stochastic gradient descent algorithm to train the network until convergence.

We apply the trained models in DLVC. The CNN-ILF is applied after deblocking filter and before sample adaptive offset. There are different models corresponding to different QPs, and one model is selected for each frame according to the frame's QP. For each CTU, there are two binary flags that control the on/off of the CNN-ILF for luma and chroma, respectively. These flags are decided at the encoder side and transmitted to the decoder.

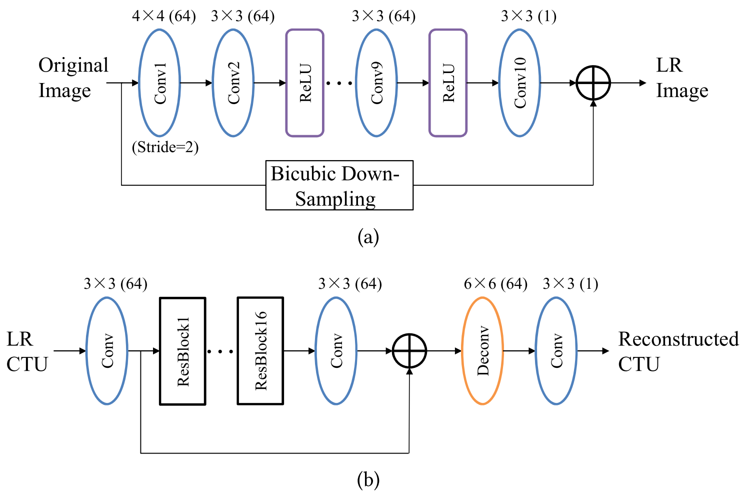


Fig. 7. The network structure of the proposed CNN-BARC, including (a) CNN-DS for down-sampling and (b) CNN-US for up-sampling. Note that the first Conv layer in (a) has a stride of 2 to achieve 2× down-sampling.

4.2 CNN-Based Block Adaptive Resolution Coding

CNN-BARC is a down- and up-sampling-based coding tool that uses trained CNN models as the down- and up-sampling filters. In DLVC, CNN-BARC is applied only for intra frame coding. The down-sampling coding or direct coding mode is decided for each CTU, and the down-sampling coding mode has two sub-modes: using CNNs for down- and up-sampling, and using simple interpolation filters for down- and up-sampling. All the modes and sub-modes are decided by the encoder and signaled to the decoder.

The networks for down- and up-sampling are illustrated in Figure 7. Specifically, the down-sampling CNN (CNN-DS) has 10 convolutional layers where the first layer has a stride of 2 to achieve 2× down-sizing. CNN-DS also embraces residue learning but here the original image is bicubic down-sampled to serve as the skip connection. The up-sampling CNN (CNN-US) is similar to the SR network in [75], and has 16 ResBlocks, 3 convolutional layers, 1 deconvolutional layer, and a global skip connection.

The CNN-DS and CNN-US are trained in four steps. First, we remove the convolutional layers in the CNN-DS, making it a simple bicubic down-sampling, and train the CNN-US to minimize the end-to-end MSE (i.e., the error between original image and down-sampled-up-sampled image). Second, we add back the layers of CNN-DS, fix the parameters of the CNN-US, and train the CNN-DS to minimize the end-to-end MSE. Third, we fine-tune the parameters of CNN-DS and CNN-US simultaneously, using a combination of two losses: the one is end-to-end MSE, and the other is the down-sampled MSE (i.e., the error between bicubic down-sampled image and network down-sampled image), where the latter serves as a regularization term. Fourth, we fix the parameters of the CNN-DS, and compress the down-sampled images by the DLVC intra coding (turning off all in-loop filters) at different QPs. For each QP, we train a separate CNN-US model.

There are two mode selection steps regarding CNN-BARC in the DLVC encoder. The first is to decide which down- and up-sampling (sub-)mode, and the second is to decide whether to perform down-sampling. We compare the rate-distortion costs of different modes to make decision. The rate is the number of coded bits, and the distortion is the MSE between original and reconstructed CTUs. For fair comparison, we always calculate the distortion at the original resolution. Last but

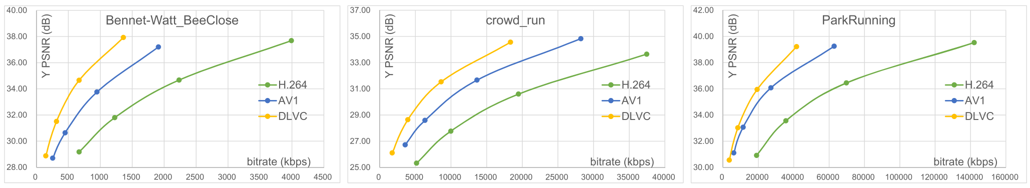


Fig. 8. Rate-quality curves of H.264, AV1, and DLVC on three videos: Bennet-Watt_BeeClose (VGA resolution), crowd_run (HD resolution), and ParkRunning (UHD resolution).

not the least, after an intra frame is compressed, we perform up-sampling again for the down-sampled-coded CTUs. More details about CNN-BARC can be found in [73, 74].

4.3 Experimental Settings

We evaluate the compression performance of DLVC on a set of 66 video sequences that have different spatial resolutions. The set consists of three subsets: 34 videos having VGA resolution (640×480) from the Consumer Digital Video Library (CDVL),[4] 22 videos having HD resolution (1920×1080) from Xiph,[5] and 10 videos recommended by JVET. The CDVL subset and Xiph subset are identical to the dataset used in [107]. The JVET subset is used for JCfP, and is further divided into Class A and Class B: Class A has 5 videos of UHD resolution (3840×2160), Class B has 5 videos of HD resolution, as described in [111].

Since DLVC was developed as a proposal, we strictly followed the test conditions described in [111], and compare DLVC with HM and JEM. We directly use the results of HM and JEM provided by JVET in JCfP [111]. The corresponding software versions are HM 16.16[6] and JEM 7.0.[7] To match the bitrate, DLVC uses the same configurations, and QP is adjusted for each sequence individually.

In addition, we compare DLVC with H.264 (using x264 encoder) and AV1. H.264 is nowadays the *de facto* must-have for ubiquitous videos, AV1 represents state-of-the-art open-source video codecs. We test the random-access configuration for comparing H.264, AV1, and DLVC, since random-access configuration usually leads to the highest compression ratio of a video codec. For H.264 and AV1, we use the FFmpeg framework with libx264 and libaom-av1 encoders, respectively. The codec version is 58.55.100.[8] The configuration of H.264 is "medium" mode, and the constant rate factor (crf) is set to 20, 25, 30, 35. The configuration of AV1 is "constant quality" mode, and crf is set to 35, 45, 55, 60. To match the quality, DLVC uses random-access configuration, and QP is set to 27, 32, 37, 42.

4.4 Experimental Results

Figure 8 presents the rate-quality curves of H.264, AV1, and DLVC on three typical videos in the test set. Here, rate is measured in kilobits per second (kbps) and quality is measured in PSNR for Y, U, V channels separately. Only Y PSNR is shown in the figure. The trends of U and V are similar to Y. Once we have such rate-quality curves, we can calculate the BD-rate between any two curves [13]: first we calculate the area between a curve and the vertical axis, then we calculate BD-rate $= (\text{Area}_{\text{test}} - \text{Area}_{\text{anchor}})/\text{Area}_{\text{anchor}}$. For example, for the ParkRunning sequence, the BD-rate of Y PSNR of DLVC (test) over H.264 (anchor) is −71.3%. That says, when achieving the same

[4]https://www.cdvl.org/.
[5]https://media.xiph.org/video/derf/.
[6]https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.16/.
[7]https://jvet.hhi.fraunhofer.de/svn/svn_HMJEMSoftware/tags/HM-16.6-JEM-7.0/.
[8]https://ffmpeg.org/.

Table 2. BD-rate Results of DLVC Over H.264 and AV1

Test subset	DLVC vs. H.264			DLVC vs. AV1		
	Y	U	V	Y	U	V
CDVL (VGA)	−78.6%	−81.4%	−79.2%	−37.3%	−48.5%	−46.5%
Xiph (HD)	−78.4%	−78.2%	−78.2%	−37.0%	−50.9%	−48.1%
JVET Class A (UHD)	−82.7%	−81.1%	−82.8%	−40.1%	−62.3%	−58.0%
JVET Class B (HD)	−84.1%	−86.4%	−86.4%	−42.9%	−52.9%	−53.6%

Table 3. BD-Rate Results of DLVC Over HM

	Random-Access			Low-Delay		
	Y	U	V	Y	U	V
FoodMarket	−38.42%	−45.57%	−50.10%	-	-	-
CatRobot	−46.87%	−59.26%	−53.80%	-	-	-
DaylightRoad	−47.34%	−60.65%	−46.19%	-	-	-
ParkRunning	−38.43%	−29.65%	−31.58%	-	-	-
CampfireParty	−41.20%	−38.07%	−54.53%	-	-	-
Avg. Class A	−42.45%	−46.64%	−47.24%	-	-	-
BQTerrace	−37.53%	−50.06%	−56.97%	−35.56%	−52.79%	−59.11%
RitualDance	−33.30%	−42.15%	−45.71%	−28.23%	−41.19%	−43.19%
MarketPlace	−35.34%	−51.78%	−49.99%	−26.85%	−46.65%	−47.83%
BasketballDrive	−37.09%	−52.86%	−51.20%	−32.98%	−50.91%	−52.69%
Cactus	−40.80%	−50.92%	−46.82%	−41.37%	−56.66%	−54.68%
Avg. Class B	−36.81%	−49.55%	−50.14%	−33.00%	−49.64%	−51.50%
Avg. JVET	**−39.63%**	**−48.10%**	**−48.69%**	**−33.00%**	**−49.64%**	**−51.50%**

quality levels (Y PSNR is between about 30 and 40 dB), DLVC saves on average 71.3% bits than H.264.

Table 2 summarizes the BD-rate results of DLVC compared to the H.264 anchor and the AV1 anchor on the entire test set. DLVC achieves on average around 70% to 80% bits saving than H.264, and on average around 30% to 40% bits saving than AV1, respectively. Thus, DLVC improves the compression efficiency very significantly. The BD-rate results across different resolutions are consistent in general, while the BD-rate on UHD videos is higher.

Table 3 and Table 4 present the results of DLVC in JCfP. In Table 3, DLVC is compared to the HM anchor on the JVET test sequences. Considering the Y channel, DLVC achieves on average 39.6% and 33.0% bits saving than HM, under random-access and low-delay configurations, respectively. Table 4 presents the BD-rate results of DLVC (document number J0032) compared to the JEM anchor on the JVET test sequences. We also include the BD-rate results of the other proposals in JCfP in the table. Considering the Y channel, DLVC achieves on average 10.1% and 11.8% bits saving than JEM, under random-access and low-delay configurations, respectively. DLVC is among the best proposals from the perspective of compression efficiency. It is worth noting that several other proposals, including J0014, J0018, J0022, and J0031, also adopted deep learning-based video coding tools.

Table 5 and Table 6 verify the effectiveness of the proposed CNN-ILF and CNN-BARC, respectively. In Table 5, we use a variant of HM that adds the quadruple-tree-binary-tree (QTBT) structure, which outperforms the vanilla HM, as the anchor. We integrate the CNN-ILF into the anchor

Table 4. BD-Rate Results of All the Proposals in JCfP Compared to JEM

Proposal	Organizations	Random-Access			Low-Delay		
		Y	U	V	Y	U	V
J0011	DJI, Peking Univ.	−1.57%	−0.71%	−1.72%	−3.30%	−0.67%	−4.26%
J0012	Ericsson, Nokia	−0.90%	−1.14%	−1.15%	−0.17%	−0.48%	−1.93%
J0013	ETRI, Sejong Univ.	0.64%	−0.39%	−0.89%	0.85%	−1.27%	−2.50%
J0014	Fraunhofer HHI	−7.55%	−6.94%	−5.96%	−7.22%	−7.62%	−5.71%
J0015	InterDigital, Dolby	−3.98%	−3.28%	−3.16%	−3.64%	−2.55%	−4.48%
J0016	KDDI	−0.57%	−0.52%	−1.30%	−0.17%	0.38%	−0.40%
J0017	LG Electronics	−2.52%	−5.29%	−6.19%	−2.09%	−2.47%	−3.70%
J0018	MediaTek	−16.60%	−6.75%	−10.43%	−9.41%	−1.92%	−3.35%
		−14.40%	−5.13%	−8.82%	−7.20%	0.23%	−0.96%
J0020	Panasonic	−2.28%	−3.44%	−3.88%	−2.02%	−1.02%	−2.36%
		−0.06%	0.91%	0.52%	−0.09%	2.87%	1.21%
J0021	Qualcomm, Technicolor	−15.53%	−3.66%	−5.97%	−12.65%	−17.40%	−19.95%
		−10.26%	0.05%	−1.65%	−9.87%	−11.73%	−14.88%
J0022		−13.60%	−3.80%	−5.63%	−12.69%	−16.65%	−18.75%
J0023	RWTH Aachen Univ.	−0.79%	−1.52%	−1.52%	−0.84%	−0.58%	−0.80%
J0024	Samsung, Huawei, GoPro, HiSilicon	−6.01%	10.34%	8.53%	−0.38%	14.73%	15.84%
		−4.24%	10.71%	9.23%	-	-	-
J0026	Sharp, Foxconn	−6.15%	−5.68%	−5.68%	−5.57%	9.00%	8.69%
J0027	NHK, Sharp	−2.14%	−5.55%	−5.61%	−2.23%	−1.97%	−3.83%
J0028	Sony	−2.41%	−4.85%	−5.14%	−2.25%	−6.74%	−7.34%
J0029	Tencent	−4.70%	−8.34%	−8.91%	−4.47%	−9.47%	−10.82%
J0031	Univ. Bristol	−4.54%	20.19%	18.68%	−0.52%	5.74%	5.81%
J0032	**USTC and others**	**−10.11%**	**−9.59%**	**−9.97%**	**−11.83%**	**−13.22%**	**−16.49%**

More details can be obtained at http://phenix.it-sudparis.eu/jvet/doc_end_user/current_meeting.php?id_meeting=174&search_id_group=1&search_sub_group=1.

Table 5. BD-Rate Results of CNN-ILF on Top of HM + QTBT

	Random-Access			Low-Delay			All-Intra		
	Y	U	V	Y	U	V	Y	U	V
Avg. Class A	−5.1%	−3.8%	−4.5%	-	-	-	−5.7%	−5.1%	−5.9%
Avg. Class B	−5.9%	−5.8%	−5.8%	−5.2%	−7.2%	−8.4%	−7.1%	−6.5%	−7.8%
Avg. JVET	**−5.5%**	**−4.8%**	**−5.1%**	**−5.2%**	**−7.2%**	**−8.4%**	**−6.4%**	**−5.8%**	**−6.8%**

Table 6. BD-Rate Results of CNN-BARC on Top of HM + QTBTTT

	Random-Access			Low-Delay			All-Intra		
	Y	U	V	Y	U	V	Y	U	V
Avg. Class A	−1.8%	0.6%	0.4%	-	-	-	−7.2%	2.0%	2.0%
Avg. Class B	−1.0%	0.1%	−0.1%	−0.3%	−0.3%	−0.0%	−3.6%	0.5%	0.4%
Avg. JVET	**−1.4%**	**0.3%**	**0.2%**	**−0.3%**	**−0.3%**	**−0.0%**	**−5.4%**	**1.3%**	**1.2%**

and turn on/off the CNN-ILF for comparison. As shown, CNN-ILF achieves significant BD-rate reduction: on average 5.5%, 5.2%, 6.4% for the Y channel under random-access, low-delay, all-intra configurations, respectively. In Table 6, the anchor is another variant of HM that adds the quadruple-tree-binary-tree-triple-tree (QTBTTT) structure, which further outperforms the HM plus QTBT. We integrate the CNN-BARC into the anchor and turn on/off the CNN-BARC for comparison. As shown, CNN-BARC achieves significant BD-rate reduction under all-intra configuration: on average 5.4% for the Y channel. The BD-rate under random-access and low-delay configurations is less significant, because CNN-BARC is applied on intra frames only.

5 PERSPECTIVES AND CONCLUSIONS

In this section, we present our perspectives on some open problems for future research, such as whether to choose deep schemes or to choose deep tools, how to pursue high compression efficiency while reducing computational complexity, and so on. We then discuss potential future work.

5.1 Open Problems

Deep Schemes versus Deep Tools. Shall we be ambitious to expect deep scheme to be the future of video coding, or shall we be satisfied with deep tools within traditional non-deep schemes? In other words, can the non-deep schemes be completely replaced by deep schemes? As for now, the answer to this question is probably "no" because deep schemes in general do not outperform non-deep schemes for video coding. But as research goes on, the answer may become "yes" via two ways: first, deep schemes may be improved so much that they clearly beat non-deep schemes; second, the coding tools in a traditional coding scheme (e.g., HEVC) may be all replaced by corresponding deep tools, leading to a "deepened" coding scheme that is better than before. The second way may be more practical according to our subjective feeling.

Compression Efficiency versus Computational Complexity. Comparing the existing deep tools with their counterparts in the traditional non-deep schemes, one may easily notice that the computational complexity of the former is much higher than the latter. High complexity is indeed a general issue of deep learning, and a critical issue that hinders the adoption of deep networks in scenarios of limited computing resource, e.g., mobile phones. This general issue is now addressed at two aspects: first, to develop novel, efficient, compact deep networks that maintain the high performance (i.e. compression efficiency for video coding) but require much less computations; second, to advocate the adoption of hardware that is specifically designed for deep networks.

Optimization for Perceptual Naturalness or Semantic Quality. Coding schemes designed for natural video are usually serving for human viewing, e.g., television, movie, micro-video. It is natural for these schemes that the quality of the reconstructed video shall be evaluated based on human perception. Nonetheless, for traditional non-deep coding schemes, the most widely adopted quality metric is still PSNR, which corresponds to human perception poorly. For deep schemes or deep tools, a few works have been done to optimize them for perceptual naturalness, e.g., using discriminator loss. Moreover, there are coding schemes that serve for automatic semantic analysis instead of human viewing, such as surveillance video coding. For these schemes, the quality metric shall be semantic quality [80], which remains largely unexplored. As a special note, we find that there is a tradeoff between signal fidelity, perceptual naturalness, and semantic quality [81], which implies that the optimization target shall be aligned with the actual requirement.

Speciality versus Universality. To one extreme, can one coding scheme be simply the best for any kind of video? The answer is "no" due to the no-free-lunch theorem, which is claimed in the machine learning literature [142] and also applies for compression. To another extreme, can we have a special coding scheme for each video? Not to mention the practical difficulty, such a coding "strategy" is useless because it is no more than assigning an identifier to each video. In between the two extremes are practical and useful coding schemes. That means, coding schemes shall have both speciality and universality to some extent. For deep schemes and deep tools, it implies that the training data shall be carefully selected to reflect the interesting data domain. More works about this aspect are expected.

Federated Design of Multiple Deep Tools. Currently, most of deep tools have been designed individually, but once they are applied jointly, they may not collaborate well or may even conflict with each other. The underlying reason is that multiple coding tools are indeed dependent. For example, different prediction tools generate different predictions and lead to a variety of residual signals, so transform tools dealing with residual signals perform differently. Ideally, multiple deep tools shall be designed in a federated manner. However, this can be difficult because the dependency among tools is complicated.

5.2 Future Work

In the predictable future, the requirement about video coding technology is still increasing. For entertainment, virtual reality and augmented reality applications are calling for techniques to tackle with new data, such as depth map, point cloud, 3D surface, and so on. For surveillance, the need of intelligent analysis pushes the upgrade of video resolution. For scientific observation, more and more observing instruments are directly connected to a video recorder and generate massive video data. All these requirements drive the development of video coding to achieve higher compression efficiency, lower computational complexity, and smarter integration into video analytical systems. We believe deep learning-based video coding techniques are promising for these challenging objectives. Especially, we expect a holistic framework based on deep networks and integrating image/video acquisition, coding, processing, analysis, and understanding, which indeed mimics human vision system.

ACKNOWLEDGMENTS

The authors would like to thank the following colleagues and collaborators: Jizheng Xu, Bin Li, Zhibo Chen, Li Li, Fangdong Chen, Yuanying Dai, Changsheng Gao, Lei Guo, Shuai Huo, Ye Li, Kang Liu, Changyue Ma, Haichuan Ma, Rui Song, Yefei Wang, Ning Yan, Kun Yang, Qingyu Zhang, Zhenxin Zhang, and Haitao Yang.

REFERENCES

[1] Mariana Afonso, Fan Zhang, and David R. Bull. 2019. Video compression based on spatio-temporal resolution adaptation. *IEEE Transactions on Circuits and Systems for Video Technology* 29, 1 (2019), 275–280.
[2] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc Van Gool. 2017. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *NIPS*. 1141–1151.
[3] Eirikur Agustsson, Michael Tschannen, Fabian Mentzer, Radu Timofte, and Luc Van Gool. 2018. Extreme learned image compression with GANs. In *CVPR Workshops*. 2587–2590.
[4] Eze Ahanonu, Michael Marcellin, and Ali Bilgin. 2018. Lossless image compression using reversible integer wavelet transforms and convolutional neural networks. In *DCC*. IEEE, 395.
[5] Mohammad Akbari, Jie Liang, and Jingning Han. 2019. DSSLIC: Deep semantic segmentation-based layered image compression. In *ICASSP*. 2042–2046.
[6] Mohammad Haris Baig, Vladlen Koltun, and Lorenzo Torresani. 2017. Learning to inpaint for image compression. In *NIPS*. 1246–1255.
[7] Mohammad Haris Baig and Lorenzo Torresani. 2017. Multiple hypothesis colorization and its application to image compression. *Computer Vision and Image Understanding* 164 (2017), 111–123.

[8] Johannes Ballé. 2018. Efficient nonlinear transforms for lossy image compression. In *PCS*. 248–252.
[9] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. 2016. End-to-end optimization of nonlinear transform codes for perceptual quality. In *PCS*. IEEE, 1–5.
[10] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. 2016. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704* (2016).
[11] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. 2018. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436* (2018).
[12] Yoshua Bengio and Samy Bengio. 2000. Modeling high-dimensional discrete data with multi-layer neural networks. In *NIPS*. 400–406.
[13] Gisle Bjontegaard. 2001. *Calculation of Average PSNR Differences between RD-curves*. Technical Report VCEG-M33. VCEG.
[14] Yochai Blau and Tomer Michaeli. 2018. The perception-distortion tradeoff. In *CVPR*. 6228–6237.
[15] Chunlei Cai, Li Chen, Xiaoyun Zhang, and Zhiyong Gao. 2019. Efficient variable rate image compression with multi-scale decomposition network. *IEEE Transactions on Circuits and Systems for Video Technology* 29, 12 (2019), 3687–3700. DOI: 10.1109/TCSVT.2018.2880492
[16] Lukas Cavigelli, Pascal Hager, and Luca Benini. 2017. CAS-CNN: A deep convolutional neural network for image compression artifact suppression. In *IJCNN*. IEEE, 752–759.
[17] Honggang Chen, Xiaohai He, Linbo Qing, Shuhua Xiong, and Truong Q. Nguyen. 2018. DPW-SDNet: Dual pixel-wavelet domain deep CNNs for soft decoding of JPEG-compressed images. In *CVPR Workshops*. 711–720.
[18] Tong Chen, Haojie Liu, Qiu Shen, Tao Yue, Xun Cao, and Zhan Ma. 2017. DeepCoder: A deep neural network based video compression. In *VCIP*. IEEE, 1–4.
[19] Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. 2018. PixelSNAIL: An improved autoregressive generative model. In *ICML*. 863–871.
[20] Yunjin Chen and Thomas Pock. 2016. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 6 (2016), 1256–1272.
[21] Zhibo Chen and Tianyu He. 2019. Learning based facial image compression with semantic fidelity metric. *Neurocomputing* 338 (2019), 16–25.
[22] Zhibo Chen, Tianyu He, Xin Jin, and Feng Wu. 2019. Learning for video compression. *IEEE Transactions on Circuits and Systems for Video Technology*. DOI: 10.1109/TCSVT.2019.2892608
[23] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. 2018. Deep convolutional autoencoder-based lossy image compression. In *PCS*. IEEE, 253–257.
[24] Michele Covell, Nick Johnston, David Minnen, Sung Jin Hwang, Joel Shor, Saurabh Singh, Damien Vincent, and George Toderici. 2017. Target-quality image compression with recurrent, convolutional neural networks. *arXiv preprint arXiv:1705.06687* (2017).
[25] Wenxue Cui, Tao Zhang, Shengping Zhang, Feng Jiang, Wangmeng Zuo, Zhaolin Wan, and Debin Zhao. 2017. Convolutional neural networks based intra prediction for HEVC. In *DCC*. IEEE, 436.
[26] Yuanying Dai, Dong Liu, and Feng Wu. 2017. A convolutional neural network approach for post-processing in HEVC intra coding. In *MMM*. Springer, 28–39.
[27] Yuanying Dai, Dong Liu, Zheng-Jun Zha, and Feng Wu. 2018. A CNN-based in-loop filter with CU classification for HEVC. In *VCIP*. 1–4.
[28] Chao Dong, Yubin Deng, Chen Change Loy, and Xiaoou Tang. 2015. Compression artifacts reduction by a deep convolutional network. In *ICCV*. 576–584.
[29] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. 2014. Learning a deep convolutional network for image super-resolution. In *ECCV*. Springer, 184–199.
[30] Robert D. Dony and Simon Haykin. 1995. Neural network approaches to image compression. *Proc. IEEE* 83, 2 (1995), 288–303.
[31] Thierry Dumas, Aline Roumy, and Christine Guillemot. 2017. Image compression with stochastic winner-take-all auto-encoder. In *ICASSP*. IEEE, 1512–1516.
[32] Thierry Dumas, Aline Roumy, and Christine Guillemot. 2018. Autoencoder based image compression: Can the learning be quantization independent?. In *ICASSP*. IEEE, 1188–1192.
[33] Longtao Feng, Xinfeng Zhang, Xiang Zhang, Shanshe Wang, Ronggang Wang, and Siwei Ma. 2018. A dual-network based super-resolution for compressed high definition video. In *PCM*. Springer, 600–610.
[34] Chih-Ming Fu, Elena Alshina, Alexander Alshin, Yu-Wen Huang, Ching-Yeh Chen, Chia-Yang Tsai, Chih-Wei Hsu, Shaw-Min Lei, Jeong-Hoon Park, and Woo-Jin Han. 2012. Sample adaptive offset in the HEVC standard. *IEEE Transactions on Circuits and Systems for Video Technology* 22, 12 (2012), 1755–1764.
[35] Leonardo Galteri, Lorenzo Seidenari, Marco Bertini, and Alberto Del Bimbo. 2017. Deep generative adversarial compression artifact removal. In *ICCV*. 4826–4835.

[36] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*. 580–587.

[37] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*. 2672–2680.

[38] Karol Gregor, Frederic Besse, Danilo Jimenez Rezende, Ivo Danihelka, and Daan Wierstra. 2016. Towards conceptual compression. In *NIPS*. 3549–3557.

[39] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. 2015. DRAW: A recurrent neural network for image generation. In *ICML*. 1462–1471.

[40] Karol Gregor and Yann LeCun. 2011. Learning representations by maximizing compression. *arXiv preprint arXiv:1108.1169* (2011).

[41] Jun Guo and Hongyang Chao. 2016. Building dual-domain representations for compression artifacts reduction. In *ECCV*. Springer, 628–644.

[42] Jun Guo and Hongyang Chao. 2017. One-to-many network for visually pleasing compression artifacts reduction. In *CVPR*. 3038–3047.

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.

[44] Xiaoyi He, Qiang Hu, Xiaoyun Zhang, Chongyang Zhang, Weiyao Lin, and Xintong Han. 2018. Enhancing HEVC compressed videos with a partition-masked convolutional neural network. In *ICIP*. IEEE, 216–220.

[45] Geoffrey Hinton and Ruslan Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313, 5786 (2006), 504–507.

[46] Jun-Hao Hu, Wen-Hsiao Peng, and Chia-Hua Chung. 2018. Reinforcement learning for HEVC/H.265 intra-frame rate control. In *ISCAS*. IEEE, 1–5.

[47] Yueyu Hu, Wenhan Yang, Mading Li, and Jiaying Liu. 2019. Progressive spatial recurrent neural network for intra prediction. *IEEE Transactions on Multimedia* 21, 12 (2019), 3024–3037. DOI: 10.1109/TMM.2019.2920603

[48] Shuai Huo, Dong Liu, Feng Wu, and Houqiang Li. 2018. Convolutional neural network-based motion compensation refinement for video coding. In *ISCAS*. 1–4.

[49] Chuanmin Jia, Shiqi Wang, Xinfeng Zhang, Shanshe Wang, Jiaying Liu, Shiliang Pu, and Siwei Ma. 2019. Content-aware convolutional neural network for in-loop filtering in high efficiency video coding. *IEEE Transactions on Image Processing* 28, 7 (2019), 3343–3356.

[50] Feng Jiang, Wen Tao, Shaohui Liu, Jie Ren, Xun Guo, and Debin Zhao. 2018. An end-to-end compression framework based on convolutional neural networks. *IEEE Transactions on Circuits and Systems for Video Technology* 28, 10 (2018), 3007–3018.

[51] J. Jiang. 1999. Image compression with neural networks–A survey. *Signal Processing: Image Communication* 14, 9 (1999), 737–760.

[52] Zhipeng Jin, Ping An, Liquan Shen, and Chao Yang. 2017. CNN oriented fast QTBT partition algorithm for JVET intra coding. In *VCIP*. IEEE, 1–4.

[53] Zhipeng Jin, Ping An, Chao Yang, and Liquan Shen. 2018. Quality enhancement for intra frame coding via CNNs: An adversarial approach. In *ICASSP*. IEEE, 1368–1372.

[54] Nick Johnston, Damien Vincent, David Minnen, Michele Covell, Saurabh Singh, Troy Chinen, Sung Jin Hwang, Joel Shor, and George Toderici. 2018. Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks. In *CVPR*. 4385–4393.

[55] Nal Kalchbrenner, Aäron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. 2017. Video pixel networks. In *ICML*. 1771–1779.

[56] Jihong Kang, Sungjei Kim, and Kyoung Mu Lee. 2017. Multi-modal/multi-scale convolutional neural network based in-loop filter design for next generation video codec. In *ICIP*. 26–30.

[57] Sungsoo Kim, Jin Soo Park, Christos G. Bampis, Jaeseong Lee, Mia K. Markey, Alexandros G. Dimakis, and Alan C. Bovik. 2018. Adversarial video compression guided by soft edge detection. *arXiv preprint arXiv:1811.10673* (2018).

[58] Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114* (2013).

[59] Jan P. Klopp, Yu-Chiang Frank Wang, Shao-Yi Chien, and Liang-Gee Chen. 2018. Learning a code-space predictor by exploiting intra-image-dependencies. In *BMVC*. 1–12.

[60] Alexander Kolesnikov and Christoph H. Lampert. 2016. Latent variable PixelCNNs for natural image modeling. *arXiv preprint arXiv:1612.08185* (2016).

[61] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*. 1097–1105.

[62] Hugo Larochelle and Iain Murray. 2011. The neural autoregressive distribution estimator. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*. 29–37.

[63] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[64] Jooyoung Lee, Seunghyun Cho, and Seung-Kwon Beack. 2018. Context-adaptive entropy model for end-to-end optimized image compression. *arXiv preprint arXiv:1809.10452* (2018).

[65] Bin Li, Houqiang Li, Li Li, and Jinlei Zhang. 2014. λ domain rate control algorithm for high efficiency video coding. *IEEE Transactions on Image Processing* 23, 9 (2014), 3841–3854.

[66] Chen Li, Li Song, Rong Xie, and Wenjun Zhang. 2017. CNN based post-processing to improve HEVC. In *ICIP*. IEEE, 4577–4580.

[67] Jiahao Li, Bin Li, Jizheng Xu, Ruiqin Xiong, and Wen Gao. 2018. Fully connected network-based intra prediction for image coding. *IEEE Transactions on Image Processing* 27, 7 (2018), 3236–3247.

[68] Ke Li, Bahetiyaer Bare, and Bo Yan. 2017. An efficient deep convolutional neural networks model for compressed image deblocking. In *ICME*. IEEE, 1320–1325.

[69] Mu Li, Shuhang Gu, David Zhang, and Wangmeng Zuo. 2018. Enlarging context with low cost: Efficient arithmetic coding with trimmed convolution. *arXiv preprint arXiv:1801.04662* (2018).

[70] Mu Li, Wangmeng Zuo, Shuhang Gu, Debin Zhao, and David Zhang. 2018. Learning convolutional networks for content-weighted image compression. In *CVPR*. 673–681.

[71] Ye Li, Bin Li, Dong Liu, and Zhibo Chen. 2017. A convolutional neural network-based approach to rate control in HEVC intra coding. In *VCIP*. IEEE, 1–4.

[72] Yue Li, Li Li, Zhu Li, Jianchao Yang, Ning Xu, Dong Liu, and Houqiang Li. 2018. A hybrid neural network for chroma intra prediction. In *ICIP*. 1797–1801.

[73] Yue Li, Dong Liu, Houqiang Li, Li Li, Zhu Li, and Feng Wu. 2019. Learning a convolutional neural network for image compact-resolution. *IEEE Transactions on Image Processing* 28, 3 (2019), 1092–1107.

[74] Yue Li, Dong Liu, Houqiang Li, Li Li, Feng Wu, Hong Zhang, and Haitao Yang. 2018. Convolutional neural network-based block up-sampling for intra frame coding. *IEEE Transactions on Circuits and Systems for Video Technology* 28, 9 (2018), 2316–2330.

[75] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. 2017. Enhanced deep residual networks for single image super-resolution. In *CVPR Workshops*. 136–144.

[76] Jianping Lin, Dong Liu, Houqiang Li, and Feng Wu. 2018. Generative adversarial network-based frame extrapolation for video coding. In *VCIP*. 1–4.

[77] Jianping Lin, Dong Liu, Haitao Yang, Houqiang Li, and Feng Wu. 2019. Convolutional neural network-based block up-sampling for HEVC. *IEEE Transactions on Circuits and Systems for Video Technology* 29, 12 (2019), 3701–3715.

[78] Dong Liu, Zhenzhong Chen, Shan Liu, and Feng Wu. 2019. Deep learning-based technology in responses to the joint call for proposals on video compression with capability beyond HEVC. *IEEE Transactions on Circuits and Systems for Video Technology*. DOI: 10.1109/TCSVT.2019.2945057

[79] Dong Liu, Haichuan Ma, Zhiwei Xiong, and Feng Wu. 2018. CNN-based DCT-like transform for image compression. In *MMM*. Springer, 61–72.

[80] Dong Liu, Dandan Wang, and Houqiang Li. 2017. Recognizable or not: Towards image semantic quality assessment for compression. *Sensing and Imaging* 18, 1 (2017), 1–20.

[81] Dong Liu, Haochen Zhang, and Zhiwei Xiong. 2019. On the classification-distortion-perception tradeoff. In *NeurIPS*. 1204–1213.

[82] Jiaying Liu, Sifeng Xia, Wenhan Yang, Mading Li, and Dong Liu. 2019. One-for-all: Grouped variation network based fractional interpolation in video coding. *IEEE Transactions on Image Processing* 28, 5 (2019), 2140–2151.

[83] Kang Liu, Dong Liu, Houqiang Li, and Feng Wu. 2018. Convolutional neural network-based residue super-resolution for video coding. In *VCIP*. 1–4.

[84] Pengju Liu, Hongzhi Zhang, Kai Zhang, Liang Lin, and Wangmeng Zuo. 2018. Multi-level wavelet-CNN for image restoration. In *CVPR Workshops*. 773–782.

[85] Zhenyu Liu, Xianyu Yu, Yuan Gao, Shaolin Chen, Xiangyang Ji, and Dongsheng Wang. 2016. CU partition mode decision for HEVC hardwired intra encoder using convolution neural network. *IEEE Transactions on Image Processing* 25, 11 (2016), 5088–5103.

[86] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. 2019. DVC: An end-to-end deep video compression framework. In *CVPR*. 11006–11015.

[87] Sihui Luo, Yezhou Yang, Yanling Yin, Chengchao Shen, Ya Zhao, and Mingli Song. 2018. DeepSIC: Deep semantic image compression. In *International Conference on Neural Information Processing*. Springer, 96–106.

[88] Changyue Ma, Dong Liu, Xiulian Peng, Li Li, and Feng Wu. 2019. Convolutional neural network-based arithmetic coding for HEVC intra-predicted residues. *IEEE Transactions on Circuits and Systems for Video Technology*. DOI: 10.1109/TCSVT.2019.2927027

[89] Changyue Ma, Dong Liu, Xiulian Peng, and Feng Wu. 2018. Convolutional neural network-based arithmetic coding of DC coefficients for HEVC intra coding. In *ICIP*. 1772–1776.

[90] Changyue Ma, Dong Liu, Xiulian Peng, Zheng-Jun Zha, and Feng Wu. 2019. Neural network-based arithmetic coding for inter prediction information in HEVC. In *ISCAS*. 1–5.

[91] Haichuan Ma, Dong Liu, Ruiqin Xiong, and Feng Wu. 2019. A CNN-based image compression scheme compatible with JPEG2000. In *ICIP*. 704–708.

[92] Li Ma, Yonahong Tian, and Tieiun Huang. 2018. Residual-based video restoration for HEVC intra coding. In *BigMM*. IEEE, 1–7.

[93] Siwei Ma, Xinfeng Zhang, Chuanmin Jia, Zhenghui Zhao, Shiqi Wang, and Shanshe Wang. 2019. Image and video compression with neural networks: A review. *IEEE Transactions on Circuits and Systems for Video Technology*. DOI: 10.1109/TCSVT.2019.2910119

[94] Xiandong Meng, Chen Chen, Shuyuan Zhu, and Bing Zeng. 2018. A new HEVC in-loop filter based on multi-channel long-short-term dependency residual networks. In *DCC*. IEEE, 187–196.

[95] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. 2018. Conditional probability models for deep image compression. In *CVPR*. 4394–4402.

[96] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. 2019. Practical full resolution learned lossless image compression. In *CVPR*. 10629–10638.

[97] David Minnen, Johannes Ballé, and George Toderici. 2018. Joint autoregressive and hierarchical priors for learned image compression. In *NIPS*. 10794–10803.

[98] David Minnen, George Toderici, Michele Covell, Troy Chinen, Nick Johnston, Joel Shor, Sung Jin Hwang, Damien Vincent, and Saurabh Singh. 2017. Spatially adaptive image compression using a tiled deep network. In *ICIP*. IEEE, 2796–2800.

[99] Vinod Nair and Geoffrey Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *ICML*. 807–814.

[100] Andrey Norkin, Gisle Bjontegaard, Arild Fuldseth, Matthias Narroschke, Masaru Ikeda, Kenneth Andersson, Minhua Zhou, and Geert van der Auwera. 2012. HEVC deblocking filter. *IEEE Transactions on Circuits and Systems for Video Technology* 22, 12 (2012), 1746–1754.

[101] Alexander G. Ororbia, Ankur Mali, Jian Wu, Scott O'Connell, William Dreese, David Miller, and C. Lee Giles. 2019. Learned neural iterative decoding for lossy image compression systems. In *DCC*. 3–12.

[102] Woon-Sung Park and Munchurl Kim. 2016. CNN-based in-loop filtering for coding efficiency improvement. In *IEEE Image, Video, and Multidimensional Signal Processing Workshop*. IEEE, 1–5.

[103] J. Pfaff, P. Helle, D. Maniry, S. Kaltenstadler, W. Samek, H. Schwarz, D. Marpe, and T. Wiegand. 2018. Neural network based intra prediction for video coding. In *Applications of Digital Image Processing XLI*, Vol. 10752. 1075213.

[104] Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. 2017. Semantic perceptual image compression using deep convolution networks. In *DCC*. IEEE, 250–259.

[105] Saurabh Puri, Sébastien Lasserre, and Patrick Le Callet. 2017. CNN-based transform index prediction in multiple transforms framework to assist entropy coding. In *EUSIPCO*. IEEE, 798–802.

[106] Oren Rippel and Lubomir Bourdev. 2017. Real-time adaptive image compression. In *ICML*. 2922–2930.

[107] Oren Rippel, Sanjay Nair, Carissa Lew, Steve Branson, Alexander G. Anderson, and Lubomir Bourdev. 2019. Learned video compression. In *ICCV*. 3454–3463.

[108] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. 2017. PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517* (2017).

[109] Shibani Santurkar, David Budden, and Nir Shavit. 2018. Generative compression. In *PCS*. IEEE, 258–262.

[110] Ionut Schiopu, Yu Liu, and Adrian Munteanu. 2018. CNN-based prediction for lossless coding of photographic images. In *PCS*. IEEE, 16–20.

[111] Andrew Segall, Vittorio Baroncini, Jill Boyce, Jianle Chen, and Teruhiko Suzuki. 2017. *Joint Call for Proposals on Video Compression with Capability Beyond HEVC*. Technical Report JVET-H1002. JVET.

[112] Claude Elwood Shannon. 1948. A mathematical theory of communication. *Bell System Technical Journal* 27, 3 (1948), 379–423.

[113] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. 2001. The JPEG 2000 still image compression standard. *IEEE Signal Processing Magazine* 18, 5 (2001), 36–58.

[114] Jake Snell, Karl Ridgeway, Renjie Liao, Brett D. Roads, Michael C. Mozer, and Richard S. Zemel. 2017. Learning to generate images with perceptual similarity metrics. In *ICIP*. IEEE, 4277–4281.

[115] Nan Song, Zhenyu Liu, Xiangyang Ji, and Dongsheng Wang. 2017. CNN oriented fast PU mode decision for HEVC hardwired intra encoder. In *GlobalSIP*. IEEE, 239–243.

[116] Rui Song, Dong Liu, Houqiang Li, and Feng Wu. 2017. Neural network-based arithmetic coding of intra prediction modes in HEVC. In *VCIP*. 1–4.

[117] Xiaodan Song, Jiabao Yao, Lulu Zhou, Li Wang, Xiaoyang Wu, Di Xie, and Shiliang Pu. 2018. A practical convolutional neural network as loop filter for intra frame. In *ICIP*. IEEE, 1133–1137.

[118] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. 2012. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on Circuits and Systems for Video Technology* 22, 12 (2012), 1649–1668.

[119] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. 2017. Memnet: A persistent memory network for image restoration. In *ICCV*. 4539–4547.

[120] Lucas Theis and Matthias Bethge. 2015. Generative image modeling using spatial LSTMs. In *NIPS*. 1927–1935.

[121] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. 2017. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395* (2017).

[122] George Toderici, Sean M. O'Malley, Sung Jin Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell, and Rahul Sukthankar. 2015. Variable rate image compression with recurrent neural networks. *arXiv preprint arXiv:1511.06085* (2015).

[123] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. 2017. Full resolution image compression with recurrent neural networks. In *CVPR*. 5306–5314.

[124] Robert Torfason, Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. 2018. Towards image understanding from deep compression without decoding. *arXiv preprint arXiv:1803.06131* (2018).

[125] Yi-Hsuan Tsai, Ming-Yu Liu, Deqing Sun, Ming-Hsuan Yang, and Jan Kautz. 2018. Learning binary residual representations for domain-specific video streaming. In *AAAI*. 7363–7370.

[126] P. N. Tudor. 1995. MPEG-2 video compression. *Electronics & Communication Engineering Journal* 7, 6 (1995), 257–264.

[127] Benigno Uria, Iain Murray, and Hugo Larochelle. 2013. RNADE: The real-valued neural autoregressive density-estimator. In *NIPS*. 2175–2183.

[128] Benigno Uria, Iain Murray, and Hugo Larochelle. 2014. A deep and tractable density estimator. In *ICML*. 467–475.

[129] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel recurrent neural networks. In *ICML*. 1747–1756.

[130] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. 2016. Conditional image generation with PixelCNN decoders. In *NIPS*. 4790–4798.

[131] Aaron van den Oord and Benjamin Schrauwen. 2014. Factoring variations in natural images with deep Gaussian mixture models. In *NIPS*. 3518–3526.

[132] Gregory K. Wallace. 1992. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics* 38, 1 (1992), xviii–xxxiv.

[133] Tingting Wang, Mingjin Chen, and Hongyang Chao. 2017. A novel deep learning-based method of improving coding efficiency from the decoder-end for HEVC. In *DCC*. IEEE, 410–419.

[134] Tingting Wang, Wenhui Xiao, Mingjin Chen, and Hongyang Chao. 2018. The multi-scale deep decoder for the standard HEVC bitstreams. In *DCC*. IEEE, 197–206.

[135] Yang Wang, Xiaopeng Fan, Chuanmin Jia, Debin Zhao, and Wen Gao. 2018. Neural network based inter prediction for HEVC. In *ICME*. IEEE, 1–6.

[136] Yingbin Wang, Han Zhu, Yiming Li, Zhenzhong Chen, and Shan Liu. 2018. Dense residual convolutional neural network based in-loop filter for HEVC. In *VCIP*. IEEE, 1–4.

[137] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. 2004. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (2004), 600–612.

[138] Zhangyang Wang, Ding Liu, Shiyu Chang, Qing Ling, Yingzhen Yang, and Thomas S. Huang. 2016. D3: Deep dual-domain based fast restoration of JPEG-compressed images. In *CVPR*. 2764–2772.

[139] Yijing Watkins, Oleksandr Iaroshenko, Mohammad Sayeh, and Garrett Kenyon. 2018. Image compression: Sparse coding vs. bottleneck autoencoders. In *IEEE Southwest Symposium on Image Analysis and Interpretation*. IEEE, 17–20.

[140] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, and Ajay Luthra. 2003. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology* 13, 7 (2003), 560–576.

[141] Ian H. Witten, Radford M. Neal, and John G. Cleary. 1987. Arithmetic coding for data compression. *Commun. ACM* 30, 6 (1987), 520–541.

[142] David H. Wolpert and William G. Macready. 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1, 1 (1997), 67–82.

[143] Chao-Yuan Wu, Nayan Singhal, and Philipp Krähenbühl. 2018. Video compression through image interpolation. In *ECCV*. 416–431.

[144] Feng Wu, Dong Liu, Jizheng Xu, Bin Li, Houqiang Li, Zhibo Chen, Li Li, Fangdong Chen, Yuanying Dai, Lei Guo, Ye Li, Yue Li, Jianping Lin, Changyue Ma, Ning Yan, Wen Gao, Siwei Ma, Ruiqin Xiong, Yiqun Xu, Jiahao Li, Xiaopeng Fan, Na Zhang, Yang Wang, Tao Zhang, Min Gao, Zhenzhong Chen, Yan Zhou, Xiang Pan, Yiming Li, Feiyang Liu, and Yingbin Wang. 2018. *Description of SDR Video Coding Technology Proposal by University of Science and Technology of China, Peking University, Harbin Institute of Technology, and Wuhan University*. Technical Report JVET-J0032. JVET.

[145] Jingyao Xu, Mai Xu, Yanan Wei, Zulin Wang, and Zhenyu Guan. 2019. Fast H.264 to HEVC transcoding: A deep learning method. *IEEE Transactions on Multimedia* 21, 7 (2019), 1633–1645.

[146] Mai Xu, Tianyi Li, Zulin Wang, Xin Deng, Ren Yang, and Zhenyu Guan. 2018. Reducing complexity of HEVC: A deep learning approach. *IEEE Transactions on Image Processing* 27, 10 (2018), 5044–5059.

[147] Ning Yan, Dong Liu, Bin Li, Houqiang Li, Tong Xu, and Feng Wu. 2018. Convolutional neural network-based invertible half-pixel interpolation filter for video coding. In *ICIP*. 201–205.

[148] Ning Yan, Dong Liu, Houqiang Li, Bin Li, Li Li, and Feng Wu. 2019. Convolutional neural network-based fractional-pixel motion compensation. *IEEE Transactions on Circuits and Systems for Video Technology* 29, 3 (2019), 840–853.

[149] Ning Yan, Dong Liu, Houqiang Li, Bin Li, Li Li, and Feng Wu. 2019. Invertibility-driven interpolation filter for video coding. *IEEE Transactions on Image Processing* 28, 10 (2019), 4912–4925.

[150] Ning Yan, Dong Liu, Houqiang Li, and Feng Wu. 2017. A convolutional neural network approach for half-pel interpolation in video coding. In *ISCAS*. IEEE, 1–4.

[151] Ren Yang, Mai Xu, Tie Liu, Zulin Wang, and Zhenyu Guan. 2019. Enhancing quality for HEVC compressed videos. *IEEE Transactions on Circuits and Systems for Video Technology* 29, 7 (2019), 2039–2054.

[152] Ren Yang, Mai Xu, Zulin Wang, and Tianyi Li. 2018. Multi-frame quality enhancement for compressed video. In *CVPR*. 6664–6673.

[153] Ke Yu, Chao Dong, Liang Lin, and Chen Change Loy. 2018. Crafting a toolchain for image restoration by deep reinforcement learning. In *CVPR*. 2443–2452.

[154] Han Zhang, Li Song, Zhengyi Luo, and Xiaokang Yang. 2017. Learning a convolutional neural network for fractional interpolation in HEVC inter coding. In *VCIP*. IEEE, 1–4.

[155] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. 2017. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing* 26, 7 (2017), 3142–3155.

[156] Qingyu Zhang, Dong Liu, and Houqiang Li. 2017. Deep network-based image coding for simultaneous compression and retrieval. In *ICIP*. IEEE, 405–409.

[157] Xiaoshuai Zhang, Wenhan Yang, Yueyu Hu, and Jiaying Liu. 2018. DMCNN: Dual-domain multi-scale convolutional neural network for compression artifacts removal. In *ICIP*. IEEE, 390–394.

[158] Yongbing Zhang, Tao Shen, Xiangyang Ji, Yun Zhang, Ruiqin Xiong, and Qionghai Dai. 2018. Residual highway convolutional neural networks for in-loop filtering in HEVC. *IEEE Transactions on Image Processing* 27, 8 (2018), 3827–3841.

[159] Yongbing Zhang, Lulu Sun, Chenggang Yan, Xiangyang Ji, and Qionghai Dai. 2018. Adaptive residual networks for high-quality image restoration. *IEEE Transactions on Image Processing* 27, 7 (2018), 3150–3163.

[160] Zhizheng Zhang, Zhibo Chen, Jianxin Lin, and Weiping Li. 2019. Learned scalable image compression with bidirectional context disentanglement network. In *ICME*. 1438–1443.

[161] Lijun Zhao, Huihui Bai, Anhong Wang, and Yao Zhao. 2019. Learning a virtual codec based on deep convolutional neural network to compress image. *Journal of Visual Communication and Image Representation* 63 (2019), 102589.

[162] Lei Zhao, Shiqi Wang, Xinfeng Zhang, Shanshe Wang, Siwei Ma, and Wen Gao. 2019. Enhanced motion-compensated video coding with deep virtual reference frame generation. *IEEE Transactions on Image Processing* 28, 10 (2019), 4832–4844.

[163] Zhenghui Zhao, Shiqi Wang, Shanshe Wang, Xinfeng Zhang, Siwei Ma, and Jiansheng Yang. 2019. Enhanced bi-prediction with convolutional neural network for high efficiency video coding. *IEEE Transactions on Circuits and Systems for Video Technology* 29, 11 (2019), 3291–3301. DOI: 10.1109/TCSVT.2018.2876399

[164] Lei Zhou, Chunlei Cai, Yue Gao, Sanbao Su, and Junmin Wu. 2018. Variational autoencoder for low bit-rate image compression. In *CVPR Workshops*. 2617–2620.

Received May 2019; revised September 2019; accepted October 2019