An End-to-End Learning Framework for Video Compression

Guo Lu, Xiaoyun Zhang, Wanli Ouyang, *Senior Member, IEEE*, Li Chen, Zhiyong Gao, and Dong Xu, *Fellow, IEEE*

Abstract—Traditional video compression approaches build upon the hybrid coding framework with motion-compensated prediction and residual transform coding. In this paper, we propose the first end-to-end deep video compression framework to take advantage of both the classical compression architecture and the powerful non-linear representation ability of neural networks. Our framework employs pixel-wise motion information, which is learned from an optical flow network and further compressed by an auto-encoder network to save bits. The other compression components are also implemented by the well-designed networks for high efficiency. All the modules are jointly optimized by using the rate-distortion trade-off and can collaborate with each other. More importantly, the proposed deep video compression framework is very flexible and can be easily extended by using lightweight or advanced networks for higher speed or better efficiency. We also propose to introduce the adaptive quantization layer to reduce the number of parameters for variable bitrate coding. Comprehensive experimental results demonstrate the effectiveness of the proposed framework on the benchmark datasets.

Index Terms—Video compression, neural network, end-to-end optimization, image compression

1 Introduction

Video compression is widely used to reduce storage and bandwidth requirements when storing and transmitting videos. It is reported that video content contributes to more than 80 percent internet traffic [1], and the percentage is expected to increase even further. Therefore, it is necessary to design an efficient video compression system and generate higher quality frames at a given bandwidth budget.

A lot of video compression standards have been proposed in the past decades. For example, H.264 [2] is the most widely used video codecs and H.265 [3] is the latest video compression standard. All these algorithms follow the hybrid coding architecture with motion-compensated prediction and residual transform coding. However, these algorithms [2], [3] rely on hand-crafted modules, e.g., block based motion estimation and discrete cosine transform (DCT), to reduce the spatial and temporal redundancies in the video sequences. Therefore, it is possible to further improve video compression performance by developing new learning based methods.

Recently, deep neural network (DNN) based auto-encoders for image compression [4], [5], [6], [7], [8], [9], [10], [11], [12], [13] have achieved comparable or even better performance than the traditional image codecs like JPEG [14], JPEG2000 [15] or BPG [16]. One possible explanation is that the DNN based image compression methods can employ the end-to-end training strategy and highly non-linear transform, which are not used in the traditional approaches. Besides, the existing methods also try to employ DNNs for video compression [17]. However, most work only replace one or two modules [18], [19], [20], [21], [22] in the traditional framework instead of optimizing the video compression system in an end-to-end fashion.

There are two major challenges for building an end-to-end video compression system. *First*, it is very difficult to build a learning based video compression system because of the complicated coding procedure. The existing learning based video compression approach [23] cannot exploit the power of end-to-end optimization and also ignore the widely used hybrid coding scheme in the traditional video codecs. Therefore, it is critical to combine the advantages of both neural networks and the hybrid framework in traditional compression. Moreover, to exploit the power of the end-to-end training strategy for the learning based compression system, the rate-distortion optimization technique is also required to optimize the whole system. *Second*, it is necessary to design a scheme to generate and compress the motion information that is tailored for video compression. Video compression methods heavily rely on motion information to reduce temporal redundancy in video sequences. A straightforward solution is to use the learning based optical flow approach to represent motion information, however, the current optical flow methods only aim at generating accurate flow maps and are often not optimal for a particular task [24]. Besides, the data volume of optical flow increases significantly when compared with motion

- *G. Lu, X. Zhang, L. Chen, and Z. Gao are with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: {luguo2014, xiaoyun.zhang, hilichen, zhiyong.gao}@sjtu.edu.cn.*
- *W. Ouyang and D. Xu are with the School of Electrical and Information Engineering, The University of Sydney, NSW 2006, Australia. E-mail: {wanli.ouyang, dong.xu}@sydney.edu.au.*

Manuscript received 4 Oct. 2019; revised 18 Mar. 2020; accepted 1 Apr. 2020. Date of publication 0 . 0000; date of current version 0 . 0000. (Corresponding author: Xiaoyun Zhang.) Recommended for acceptance by T. Hassner. Digital Object Identifier no. 10.1109/TPAMI.2020.2988453

information in the traditional block based compression systems. Therefore, instead of using the traditional differential methods in [2], [3], optical flows should be compressed more efficiently.

In this paper, we propose the first end-to-end deep video compression (DVC) model. Our framework combines the advantages of both neural networks and the traditional video compression methods. The contributions of this work can be summarized as follows:

- All components in video compression, i.e., motion estimation, motion compensation, residual compression, motion compression, and bit rate estimation, are implemented with the end-to-end neural networks.
- The components in the video compression system are jointly optimized based on rate-distortion tradeoff through a single loss function, which leads to higher compression efficiency.
- The proposed framework is very flexible and two variants (DVC_Lite and DVC_Pro) of our DVC framework are also proposed for speed/efficiency priority.
- We propose the adaptive quantization layer for the DVC framework, which significantly reduces the number of parameters for variable bitrate coding.
- Experimental results show that our framework outperforms the widely used video codec H.264 and the existing learning based video codec.

This work builds upon the preliminary conference paper [25] with the following substantial improvements. (1) A more efficient motion estimation approach and a lightweight motion compression network are employed in our framework to effectively generate and compress the motion information with fewer trainable parameters. Based on these techniques, the newly proposed framework in this work (named as DVC_Lite) achieves comparable performance with the DVC model in [25], while DVC_Lite reduces the FLOPs by 76 percent and is 2.2 times faster in terms of speed. (2) Due to the high flexibility of the proposed framework, an advanced model DVC_Pro is further proposed by using more efficient residual/motion compression networks and the corresponding refinement networks. When compared with the previous DVC model, DVC_Pro outperforms DVC by up to 0.7dB. (3) An adaptive quantization layer is proposed for the variable bitrate coding and reduces the number of parameters significantly. (4) More experiments and extensive analysis, including computational complexity and loss function, are provided to demonstrate the effectiveness of our proposed framework.

2 Related Work

2.1 Image Compression

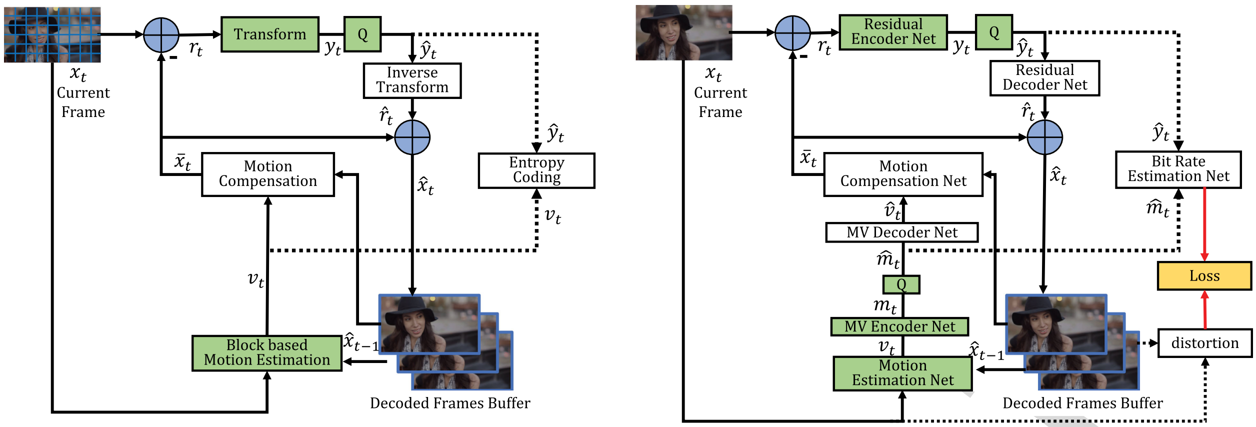
Several image compression standards [14], [15], [16] were proposed in the literature. Although these methods can compress the images effectively, they heavily rely on handcrafted techniques. For example, the JPEG standard [14] linearly maps the pixels to another domain by using DCT and the corresponding coefficients are quantized before entropy coding [14]. One disadvantage is that all the modules in the traditional codecs are separately optimized and may not achieve optimal compression performance.

Recently, the learning based image compression approaches [4], [5], [6], [7], [8], [10], [11], [11], [12], [13], [26], [27], [28], [29], [30], [31] have attracted increasing attention. In [4], [6], [9], recurrent neural networks (RNNs) based auto-encoders are utilized to design a progressive image compression scheme. And then this approach is further improved by using more advanced RNN architectures, learning based entropy model and spatial adaptive bitrate allocation [6], [9]. Other methods employed the CNNs to build an auto-encoder style network for image compression [5], [8], [10]. Besides, to optimize the learning based compression system, the methods in [4], [6], [9] only tried to minimize the distortion (e.g., mean square error) between the original frames and the reconstructed frames without considering the number of bits used for compression. Meanwhile, the rate-distortion optimization technique [32] was adopted in [5], [8], [10], [11] for higher compression efficiency by introducing the number of bits in the optimization procedure. To estimate the bit rates, the context models are learned for adaptive arithmetic coding in [11], [12], [26], while non-adaptive arithmetic coding is used in [5], [10]. In addition, other techniques such as generalized divisive normalization (GDN) [5], multi-scale image decomposition [12], adversarial training [12], importance map [11], [26], conditional probability models [26], auto-regressive model [27], [30] and intra prediction [33], [34] were proposed to improve the image compression performance. Although the learning based image compression methods outperform the traditional image codecs, it only reduces the spatial redundancy without considering the temporal relationship.

For image compression, how to generate visually pleasing reconstructed images is a critical problem. In [13], Agustsson *et al.* used the generative adversarial network based image codec to obtain high perceptual quality images at very low bitrates. Patel *et al.* [35] analyzed the commonly used metrics like PSNR or MS-SSIM and perform the human study on perceptual similarity for different image compression techniques. In [29], a deep perceptual metric was proposed for learning based image compression by using better quality for human eyes. Patel *et al.* [31] proposed the rate-distortion-perception trade-off by introducing the perception term in the optimization procedure.

2.2 Video Compression

Recently, deep learning techniques are employed for video compression [17]. Most methods aim to improve the performance of the existing video compression algorithms by replacing particular modules, such as intra prediction and residual coding [18], mode decision [19], entropy coding [20] and post-processing [21], [22]. However, these methods are not optimized in an end-to-end fashion. In [36], Chen *et al.* proposed a block based learning approach for video compression. However, it will inevitably generate blockiness artifact in the boundary between blocks. Furthermore, they used the motion information propagated from the previously reconstructed frames through the traditional block based motion estimation method, which will degrade the compression performance. Tsai *et al.* proposed an autoencoder network to compress the residual from the H.264 encoder for the specific domain videos [37]. This work does



(a) Traditional Video Compression Framework (b) Our End-to-end Video Compression Framework

Fig. 1. (a): The predictive coding architecture used by the traditional video codec H.264 [2] or H.265 [3]. (b): The proposed end-to-end video compression network. The modules with green color are not included in the decoder. "MV Encoder Net" and "MV Decoder Net" represent the "Motion Vector Encoder Net" and "Motion Vector Decoder Net".

not use a deep model for motion estimation, motion compensation or motion compression. In [38], Cheng *et al.* obtained the predicted frame through frame interpolation without encoding motion information, which may degrade the compression performance.

The most related work is the RNN based approach in [23], where video compression is formulated as frame interpolation. However, the motion information in their approach is also generated by the traditional block based motion estimation method, which is encoded by the existing non-deep learning based image compression method [39]. In other words, estimation and compression of motion information are not accomplished by deep models and jointly optimized with other components. Besides, the video codec in [23] only aims at minimizing the distortion (i.e., mean square error) between the original frame and the reconstructed frame without considering rate-distortion trade-off in the training procedure. In comparison, in our network, motion estimation and compression are achieved by DNNs, which is jointly optimized with other components by considering the rate-distortion trade-off of the whole compression system.

2.3 Motion Estimation

Motion estimation is a critical component in the video compression system. To reduce the computational complexity of the motion estimation procedure, the traditional video codecs use the block based motion estimation algorithms [40], [41], [42], which well support hardware implementation. However, the block based methods may introduce inaccurate motion information and thus degrade the compression performance.

In the computer vision tasks, optical flow is widely used to exploit the temporal relationship. Recently, a lot of learning based optical flow estimation methods [43], [44], [45], [46], [47] have been proposed. These approaches motivate us to integrate optical flow estimation into our end-to-end learning framework. When compared with the block based motion estimation method in the existing video compression approaches, learning based optical flow estimation methods can provide accurate motion information at pixel-level, which can be also optimized in an end-to-end manner.

It should be mentioned that the optical flow methods in [43], [44], [45], [46], [47] are designed for tracking true motion trajectory instead of considering the rate-distortion balance in video compression. Besides, due to the increased data volume, more bits are required to compress motion information if optical flow values are encoded by the traditional video compression approaches. Therefore, it is necessary to design an efficient motion compression scheme for learning based video compression methods.

3 Overview of the Proposed Framework

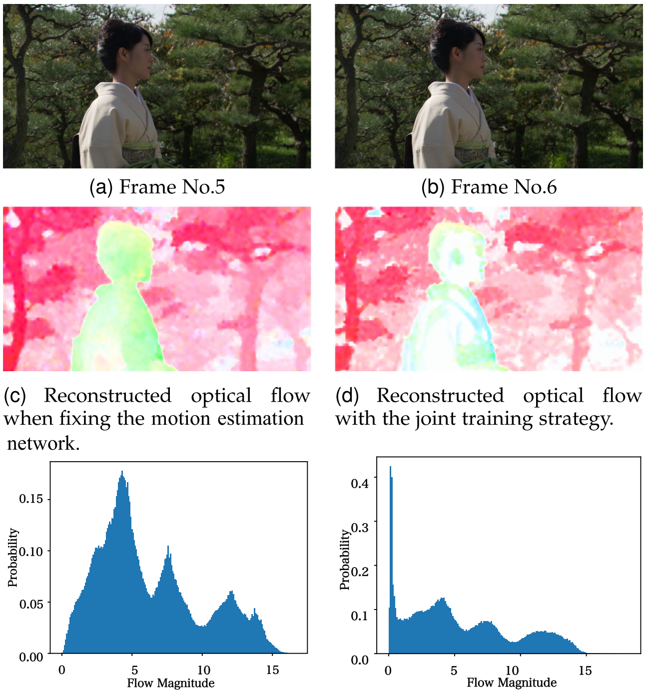
Introduction of Notations. Let $\mathcal{V} = \{x_1, x_2, \ldots, x_{t-1}, x_t, \ldots\}$ denote the current video sequences, where x_t is the frame at time step t. The predicted frame is denoted as $\bar{x}_t$ and the reconstructed/decoded frame is denoted as $\hat{x}_t$. r_t represents the residual (error) between the original frame x_t and the predicted frame $\bar{x}_t$. $\hat{r}_t$ represents the reconstructed/decoded residual. To reduce temporal redundancy, motion information is required. Among them, v_t represents the motion vector or optical flow value. And $\hat{v}_t$ is its corresponding reconstructed version. Linear or nonlinear transform can be used to improve compression efficiency. Therefore, residual information r_t is transformed to y_t, and motion information v_t can be transformed to m_t, $\hat{y}_t$ and $\hat{m}_t$ are the corresponding quantized versions, respectively.

3.1 Hybrid Coding Framework of Video Compression

In this section, we first give a brief overview of the hybrid coding framework for video compression. Please refer to [2], [3] for more details.

The hybrid coding framework is shown in Fig. 1a. All the modules in Fig. 1a are included in the encoder while green color modules are not included in the decoder. Specifically, the input frame x_t is split into a set of blocks, i.e., square regions, of the same size (e.g., 64×64). The blocks in the whole frame are encoded in raster-scan order. The encoding procedure of the traditional video compression algorithm is summarized as follows,

Step 1. Block based motion estimation. We estimate the motion between the current frame x_t and the previous



(a) Frame No.5

(b) Frame No.6

(c) Reconstructed optical flow when fixing the motion estimation network.

(d) Reconstructed optical flow with the joint training strategy.

(e) Magnitude distribution of the optical flow map (c).

(f) Magnitude distribution of the optical flow map (d).

Fig. 2. Optical flow visualization and statistic analysis.

reconstructed frame $\hat{x}_{t-1}$. The corresponding motion vector v_t for each block is obtained.

Step 2. Motion compensation. Based on the motion vector v_t from Step 1, the predicted frame $\bar{x}_t$ is obtained by copying the corresponding pixels in the previous reconstructed frame to the current frame. The residual r_t between the original frame x_t and the predicted frame $\bar{x}_t$ is obtained as $r_t = x_t - \bar{x}_t$.

Step 3. Transform and quantization. The residual r_t is first converted to a compact domain by a linear transform and then quantized to $\hat{y}_t$ for entropy coding.

Step 4. Inverse transform. The quantized result $\hat{y}_t$ in Step 3 is used by the inverse transform for obtaining the reconstructed residual $\hat{r}_t$.

Step 5. Entropy coding. Both the motion vector v_t in Step 1 and the quantized result $\hat{y}_t$ in Step 3 are encoded into bits by the entropy coding method and sent to the decoder.

Step 6. Frame reconstruction. The reconstructed frame $\hat{x}_t$ is obtained by adding $\bar{x}_t$ from Step 2 and $\hat{r}_t$ from Step 4, i.e., $\hat{x}_t = \hat{r}_t + \bar{x}_t$. The reconstructed frame will be stored in the decoded frame buffer and used for the $(t+1)$th frame at Step 1 for motion estimation.

3.2 Overview of the Proposed End-to-end Deep Video Compression

Fig. 1b provides a high-level overview of our end-to-end video compression framework. Our model follows the hybrid coding framework of motion-compensated prediction and residual transform coding, but all modules in our approach are designed and implemented by using convolution networks. The differences between our method and the traditional video compression codecs are summarized as follows,

Step 1. Motion Estimation and Compression. Instead of using the traditional block based motion estimation, we use a CNN model [44] to estimate the optical flow map, which is considered as the motion information v_t. Furthermore, in contrast to the compression method for motion information in the traditional codecs, a CNN model is proposed to compress the optical flow map in a lossy way, as the data volume of optical flow increases significantly in our work. Specifically, v_t is compressed by an auto-encoder style network with quantization, and the corresponding latent representations before and after quantization are denoted as m_t and $\hat{m}_t$, respectively. The reconstructed motion information is denoted as $\hat{v}_t$. The details are given in Section 4.2.

Step 2. Motion Compensation. Instead of using block based motion compensation as H.264/H.265, a pixel-wise motion compensation approach is implemented by using a neural network and we can then obtain the predicted frame $\bar{x}_t$ based on the optical flow map $\hat{v}_t$ obtained in Step 1. More information is provided in Section 4.3.

Step 3-4. Transform, Quantization and Inverse Transform. We replace the linear transform in the traditional compression method by using a highly non-linear residual encoder-decoder network, and the residual r_t is non-linearly mapped to the representation y_t. Then y_t is quantized to $\hat{y}_t$. The quantized representation $\hat{y}_t$ is fed into the residual decoder network to obtain the reconstructed residual $\hat{r}_t$. The details are presented in Section 4.4 and Section 5.

Step 5. Entropy Coding. At the testing stage, the quantized motion representation $\hat{m}_t$ from Step 1 and the residual representation $\hat{y}_t$ from Step 3 are encoded into bits by using arithmetic coding. At the training stage, to estimate bit cost in our proposed approach, we use the bit rate estimation net in Fig. 1 to obtain the probability distribution of each symbol in the quantized representations and calculate the entropy to approximate the bit cost. More information is provided in Section 5.

Step 6. Frame Reconstruction. The reconstructed frame $\hat{x}_t$ is generated based on the predicted frame $\bar{x}_t$ and the reconstructed residual $\hat{r}_t$.

In summary, motion information, residual information and entropy bits for hybrid coding are all learned by using networks in the proposed framework. Finally, all these functional modules are jointly optimized in an end-to-end way by using a single rate-distortion loss. And thus these modules can collaborate with each other during optimization and it is expected to achieve better compression performance. The experimental results and ablation studies validate the advantage of such an end-to-end framework. Moreover, our framework is very flexible and all network modules can be easily updated or replaced by using lightweight or advanced networks for higher speed or better performance.

4 Network Architectures for Deep Video Compression

In our proposed deep video compression framework, all the components are implemented by deep neural networks. In this section, we will introduce the motion estimation net, motion vector encoder/decoder net, motion compensation, residual encoder/decoder net and bit rate estimation net.

4.1 Motion Estimation Net

In our proposed DVC model, we use the learning based optical flow method Spynet [44] to estimate motion

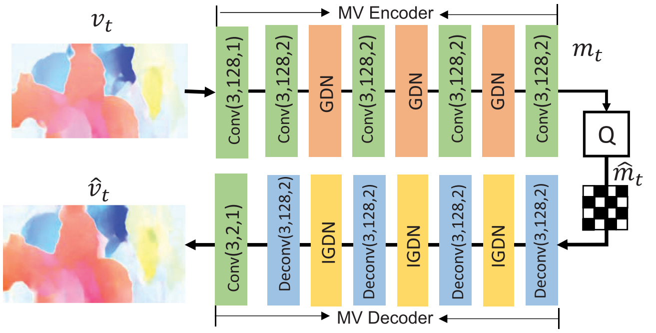
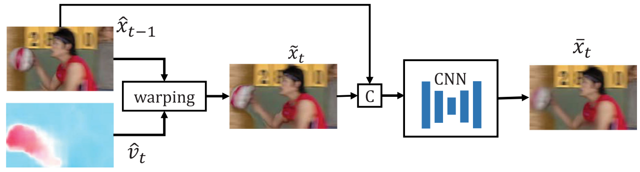
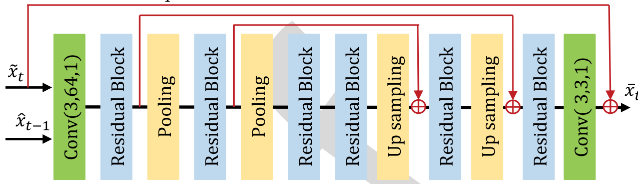


Fig. 3. Our MV Encoder-decoder network. Conv(3,128,2) represents the convolution operation with the kernel size of 3×3, the output channel of 128 and the stride of 2. GDN/IGDN [5] is the nonlinear transform function. The binary feature map is only used for illustration.



(a) Overall structure of our motion compensation net. C represents the concatenation operation.



(b) Network architecture of the CNN.

Fig. 4. Our motion compensation network.

information. Spynet employs a pyramid architecture to estimate the optical flow map between two neighboring frames in a coarse to fine manner. Specifically, the previous frame $\hat{x}_{t-1}$ and the current frame x_t are the input for Spynet and the output is the estimated optical flow map v_t, which will be compressed by the following motion compression module. In fact, any new state-of-the-art optical flow estimation net can also be adopted in the proposed compression framework and its performance improvement will benefit the whole compression framework.

Furthermore, the motion estimation net is jointly optimized with the whole compression system by minimizing the rate-distortion trade-off. Therefore, when compared with the original flow map from Spynet, the estimated flow map in our method is more compressible. In Fig. 2, we provide a visual comparison between the flow maps with or without joint training. Figs. 2a and 2b represent the frame 5 and frame 6 from the *Kimono* sequence in the HEVC Class B dataset. Fig. 2c denotes the reconstructed optical flow map when the optical flow network is fixed during the training procedure. Fig. 2d represents the reconstructed optical flow map after using the joint training strategy. Figs. 2e and 2f are the corresponding probability distributions of the optical flow magnitudes. It can be observed that the reconstructed optical flow map by joint training has more pixels with zero value, especially for the homogeneity regions like the human body in Fig. 2d. More importantly, the optical flow map with more zero values requires much fewer bits for encoding. For example, it needs 0.045 bpp for encoding the optical flow map in Fig. 2c, while it saves 15 percent bits and only needs 0.038 bpp for encoding the optical flow map in Fig. 2d.

When compared with the traditional motion estimation approaches, the learning based optical flow maps provide accurate motion and can be optimized with the whole video compression system.

4.2 MV Encoder Net and MV Decoder Net

To compress pixel-level optical flow v_t from the motion estimation network, we utilize an auto-encoder style network, which is first proposed by [5] for the image compression task. The whole Motion Vector (MV) compression network is shown in Fig. 3. The optical flow map v_t is fed into a series of convolution operations and nonlinear transform. The number of output channels for convolution (deconvolution) is 128 except for the last deconvolution layer, which is equal to 2. Given the optical flow v_t with the size of $M \times N \times 2$, the MV encoder will generate the motion representation m_t with the size of $M/16 \times N/16 \times 128$. Then m_t is quantized to $\hat{m}_t$. The MV decoder receives the quantized representation and reconstructs motion information $\hat{v}_t$. Besides, the quantized representation $\hat{m}_t$ will be used for entropy coding. Based on the proposed mv encoder and decoder network, the optical flow map can be efficiently compressed.

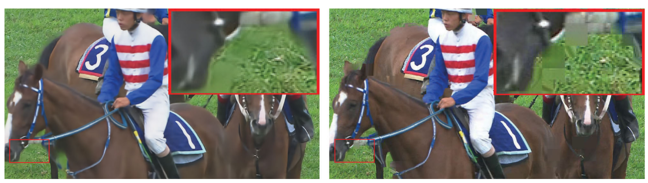
4.3 Motion Compensation Net

Given the previous reconstructed frame $\hat{x}_{t-1}$ and the motion vector $\hat{v}_t$, the motion compensation network obtains the predicted frame $\bar{x}_t$, which is expected to be as close as to current frame x_t. First, the previous frame $\hat{x}_{t-1}$ is warped to the current frame based on the motion information $\hat{v}_t$ in the following way,

$$\tilde{x}_t = \mathcal{W}(\hat{x}_{t-1}, \hat{v}_t), \tag{1}$$

where $\tilde{x}_t$ is the warped frame and $\mathcal{W}$ is the backward warp operation [48]. To further improve the quality of the warped frame $\tilde{x}_t$, we concatenate $\tilde{x}_t$ and the reference frame $\hat{x}_{t-1}$ as the input, then feed them into another CNN to obtain the refined predicted frame $\bar{x}_t$. Our motion compensation approach is shown in Fig. 4a and the detailed network architecture is provided in Fig. 4b.

Figs. 5a and 5b represent the predicted frame from our DVC model and H.265, respectively. Since our proposed method is a pixel-wise motion compensation approach, it can provide more accurate temporal information and avoid the blockiness artifact in the traditional block based motion compensation method. It means that we do not need the



(a) DVC model (b) H.265

Fig. 5. Visual comparison of the predicted frames between our model and H.265.

hand-crafted loop filter or the sample adaptive offset technique [2], [3] for post-processing.

4.4 Residual Encoder and Decoder Net

After motion estimation and motion compensation, we can obtain the predicted frame $\bar{x}_t$ and the corresponding residual information r_t. Then the residual encoder is used to map r_t to a compact domain for high efficiency compression, while the residual decoder is used to reconstruct the corresponding reconstructed residual information $\hat{r}_t$. We follow the variational image compression framework [8] to compress the residual. Specifically, the residual information is compressed based on an auto-encoder style network, which is composed of several convolution layers and GDN/IGDN layers [5]. More importantly, to provide an accurate probability estimation for the latent representation, a prior network is employed to predict the probability distribution of each representation. Please refer to [8] for more details. Compared with discrete cosine transform in the traditional video compression system, our approach can better exploit the power of non-linear transform and achieve higher compression efficiency.

4.5 Bit Rate Estimation Net

To optimize the whole network by considering both distortion and the number of bits, we need to obtain the bit rate of the generated latent representations. As we know, the accurate measure for bitrate is the entropy of the corresponding latent representation symbols. Therefore, we use the CNN model in [8] to estimate the probability distributions of $\hat{y}_t$ and $\hat{m}_t$. The bit rate estimation net in [8] employs a univariate non-parametric density model based on the cumulative to estimate the probability distribution.

5 Network Optimization

To build an end-to-end deep video compression system, it is still required to solve several issues before putting all the neural networks together. In this section, we will introduce the loss function, quantization and decoded frame buffer, which are indispensable for the end-to-end training.

5.1 Loss Function

The goal of our video compression framework is to minimize the number of bits used for encoding, while at the same time reduce the distortion between the original input frame x_t and the reconstructed frame $\hat{x}_t$. Therefore, we solve the following rate-distortion optimization problem,

$$\mathcal{L} = \lambda D + R = \lambda d(x_t, \hat{x}_t) + [H(\hat{m}_t) + H(\hat{y}_t)], \tag{2}$$

where $d(x_t, \hat{x}_t)$ denotes the distortion between x_t and $\hat{x}_t$ and can be measured by mean square error (MSE) or multi-scale structure similarity (MS-SSIM) [49]. $H(\cdot)$ represents the number of bits used for encoding the representations. In our approach, both residual representation $\hat{y}_t$ and motion representation $\hat{m}_t$ should be encoded into the bitstreams. λ determines the trade-off between the number of bits and the distortion. To stabilize the training procedure, we also introduce an auxiliary loss, which is formulated as follows,

$$\mathcal{L}_0 = \lambda D + R = \lambda[d(x_t, \hat{x}_t) + \beta d(x_t, \tilde{x}_t)] + [H(\hat{m}_t) + H(\hat{y}_t)], \tag{3}$$

where $\tilde{x}_t$ is the warped frame based on the reconstructed optical flow. The weight parameter β is set to 0.1.

5.2 Quantization

Latent representations such as residual representation y_t and motion representation m_t are required to be quantized before entropy coding. However, the quantization operation is not differential, which makes end-to-end training impossible. To address this problem, a lot of methods have been proposed [4], [5], [7]. Inspired by [5], we also replace the quantization operation by adding uniform noise in the training stage. Taking y_t as an example, the quantized representation $\hat{y}_t$ in the training stage is approximated by adding uniform noise to y_t, i.e., $\hat{y}_t = y_t + \eta$, where η is uniform noise. In the inference stage, we use the rounding operation directly, i.e., $\hat{y}_t = round(y_t)$.

5.3 Decoded Frame Buffer

As shown in Fig. 1, the previous reconstructed frame $\hat{x}_{t-1}$ is required in the motion estimation and motion compensation networks when compressing the current frame x_t. Therefore, the encoding procedure forms a chain of dependency.

To solve this issue and simplify the training procedure, we adopt an online updating strategy. Specifically, the reconstructed frame $\hat{x}_t$ in each iteration will be saved in a buffer. In the subsequent iterations, $\hat{x}_t$ in the buffer will be used for motion estimation and motion compensation when encoding x_{t+1}. Therefore, each training sample in the buffer will be updated in an epoch. In this way, we can store one previous frame for each video clip at each iteration, which is more efficient.

5.4 Training Strategy

In our implementation, we first optimize the whole network based on the loss function $\mathcal{L}_0$ for 0.5×10^6 steps. Then we use the loss function $\mathcal{L}$ for 2×10^6 steps. We use the Adam optimizer [50] by setting the initial learning rate as 0.0001. The learning rate is divided by 10 when the loss becomes stable. The motion estimation module is initialized with the pre-trained weights in [44]. The whole system is implemented based on Tensorflow and it takes about 4 days to train the whole network when using two GTX 1080Ti GPUs.

6 Extensions for Our DVC Model

Our framework is very flexible and the existing algorithms (e.g., the optical flow estimation and image compression methods) can be readily plugged into our framework. Considering that the compression efficiency and computational complexity are the two most important metrics for practical video codecs, two variants DVC_Lite and DVC_Pro of the DVC model are proposed in this section. Specifically, the DVC_Lite method is a lightweight video compression approach, which reduces 76 percent FLOPs and achieves similar compression performance when compared with the DVC model. The DVC_Pro model is an advanced video compression framework and achieves the competitive compression performance when compared with H.265 in terms of PSNR. More

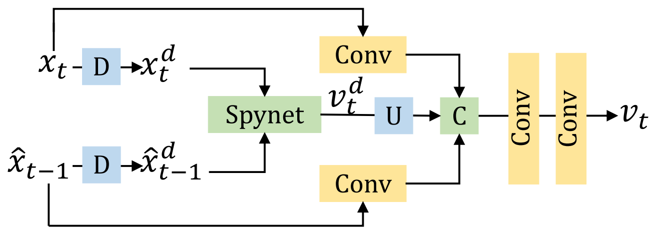


Fig. 6. The proposed motion estimation scheme. “D” and “U” represent the downsampling and upsampling operations, respectively. “C” represents the concatenation operation. “Conv” is the convolution operation.

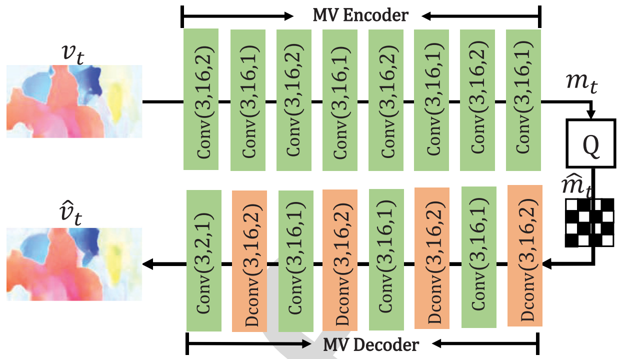


Fig. 7. Our MV Encoder-decoder network. Conv(3,16,2) represents the convolution operation with the kernel size of 3×3, the output channel of 16 and the stride of 2.

importantly, we propose the adaptive quantization layer for the learning based video compression method. Therefore, the encoder can share the same baseline model at different bitrates, which reduces the model sizes significantly.

6.1 DVC_Lite

The lightweight model DVC_Lite is based on the previous model and has the following improvements.

Motion Estimation. First, we propose an efficient motion estimation approach. Although Spynet [44] can provide accurate pixel-level optical flow, its computational complexity is also very high. In our DVC_Lite framework, we balance the accuracy of optical flow estimation and computational complexity and propose a new optical flow estimation approach to obtain the motion. Specifically, we first downsample the current frame x_t and the reference frame $\hat{x}_{t-1}$ and obtain the corresponding low resolution frames x_t^d and $\hat{x}_{t-1}^d$. Then we feed these two frames into Spynet [44] and calculate the flow information v_t^d. Finally, v_t^d is upsampled based on the context from high-resolution frames to obtain the pixel-wise motion v_t. The architecture of our motion estimation network is illustrated in Fig. 6. Instead of estimating the full resolution optical flow map directly, our scheme reduces the computational complexity significantly by performing motion estimation on the downsampled frames. More details will be provided in Section 7.

MV Encoder and Decoder Network. In our DVC_Lite framework, a lightweight motion compression framework is utilized. The new MV compression network is shown in Fig. 7. We feed the optical flow map to a series of convolution operations. The number of channels is set to 16 for all the convolution(deconvolution) layers, except for the last layer, which is set to 2. When compared with our design in Fig. 3, we increase the number of layers but greatly reduce the number of channels, our new framework thus reduces the computational complexity and model size significantly.

Motion Compensation Network and Residual Compression Network. To further reduce the computational complexity, it is also required to employ efficient network architecture for motion compensation and residual compression. A straightforward approach is to reduce the number of channels in the corresponding networks. This simple strategy is very effective to alleviate the computational complexity while maintaining the compression performance. For example, the number of channels in [8] is set to 128 while our DVC_Lite uses a light version by setting this number to 64. One possible explanation is that the residual r_t itself is very sparse and even the lightweight network can well compress the residual information.

6.2 DVC_Pro

Residual Compression. In the basic DVC model introduced in our conference work [25], the image compression framework in [8] is utilized for residual compression. To further improve the coding performance, a more advanced image compression scheme [27] is employed. In [27], Minnen *et al.* used auto-regressive and hierarchical priors to improve the efficiency of entropy coding. Due to the high flexibility of the proposed deep video compression framework, it is also straightforward to embed the learning based image compression method [27] into our framework by replacing the corresponding residual encoder and decoder.

Motion Compression. In our previous conference work [25], motion information is compressed by using the factorized entropy model, which ignores the spatial relationship in the compressed latent space. Therefore, motion compression is not very effective. When motion information occupies more percentages in the total bitrate (i.e., the low bitrate setting in Figs. 11 and 12), the performance of our DVC method drops significantly. Inspired by the entropy model in image compression [27], we also use the auto-regressive model to compress the optical flow information. As shown in Section 7.3 (see Fig. 19), the new DVC_Pro method achieves much better performance than the basic DVC network because the percentage of bits used to encode motion information drops obviously, which demonstrates that the new entropy model improves the compression performance at low bitrates.

Motion and Residual Refinement. In the DVC model, the reconstructed optical flow map and residual information can be obtained by using the MV decoder net and residual decoder net, respectively. As we know, the quantization procedure will introduce quantization errors, which means the reconstructed optical flow and residual information are not accurate and thus have compression artifacts. To further improve the compression performance, we use two motion and residual refinement modules to obtain more accurate motion and residual information. These two modules are integrated into the DVC framework, which are after the MV decoder net and the residual decoder net, respectively. The network architecture of the motion refinement network is illustrated in Fig. 8, where $\hat{v}_t'$ is the refined motion information used for the following motion compensation procedure. The network architecture of the residual refinement module is the same as the motion compensation network in Fig. 4 except that the residual information is used as the input to the network.

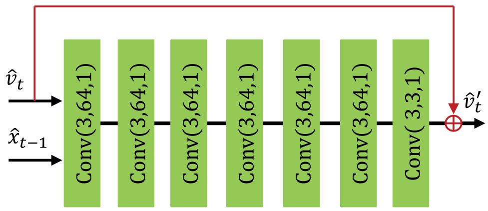


Fig. 8. The network architecture of our motion refinement network.

TABLE 1
The Network Architecture of the Proposed Adaptive Quantization Layer

Layer1	Layer2	Layer3	Layer4
Conv(1,C,1)	DepthConv(5,C,1)	DepthConv(5,C,1)	Conv(1,M,1)

Conv(1,C,1) represents the convolution layer with the kernel size of 1×1, the output channel number of C and the stride of 1. Here, M is the channel number of the compressed features in the DVC method and is equal to 192.

6.3 Adaptive Quantization Layer

In our conference work [25], we have to train different models at different λ values in Eq. (2) to achieve multiple bitrate coding in the practical applications. It means the decoder/encoder needs to store multiple models, which significantly increases the storage burden when deploying the proposed method. In this section, we aim to reduce the number of models for video coding at multiple bitrates by introducing the adaptive quantization layer (AQL).

In Fig. 9, we provide an example to illustrate how to integrate our adaptive quantization layer to the existing image compression framework [8]. The encoder (*resp. decoder*) can be the residual encoder (*resp.* decoder) or the motion encoder (*resp.* decoder) in our DVC framework. AQL (*resp.* IAQL) represents the adaptive quantization layer (*resp.* inverse adaptive quantization layer). AE and AD represent entropy encoder and entropy decoder. The hyper encoder and hyper decoder are used to estimate the corresponding entropy parameters $\hat{\sigma}$, which are used in entropy coding. In our implementation, we follow the same strategy in Fig. 9 to integrate the adaptive quantization layer to the motion compression module and residual compression module in our DVC framework.

Specifically, in our proposed video compression scheme, we first train a DVC model at high-bitrate without using the adaptive quantization layer. Then all the parameters in the baseline DVC model will be fixed after the training stage. To obtain other models at low-bitrates, we integrate the adaptive quantization layer to the pre-trained baseline DVC model at high-bitrate (see Fig. 9). Finally, we only fine-tune the adaptive quantization layers for other models at different bitrates. Therefore, we only need to store the baseline DVC model at high-bitrate and the corresponding adaptive quantization layers at low-bitrates. In other words, the models at different bitrates share the same baseline DVC model, which reduces the total size of models for multiple bitrates.

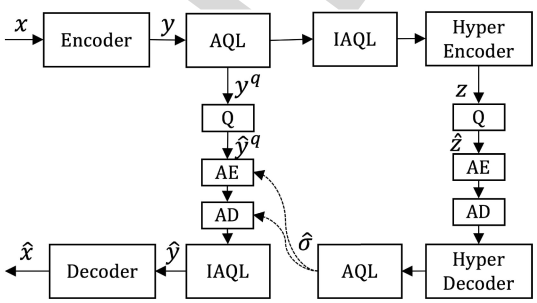


Fig. 9. An example of the basic compression network after using the adaptive quantization layer. "AQL" and "IAQL" represent the adaptive quantization layer and inverse adaptive quantization layer. "AE" and "AD" represent the arithmetic encoder and arithmetic decoder.

We use a simple network architecture to implement the adaptive quantization layer. Take Fig. 9 as an example, the compressed features y from the encoder go through the adaptive quantization layer before the actual quantization procedure. Specifically, we use a four-layer network $s(\cdot)$ to extract the scale information for the adaptive quantization layer (AQL). Then the feature y^q after adaptive quantization is defined as follows,

$$y^q = y \cdot (1 - sigmoid(s(y))). \tag{4}$$

The corresponding inverse adaptive quantization layer (IAQL) is formulated as,

$$\hat{y} = \hat{y}^q \cdot (1 + Relu(\bar{s}(\hat{y}^q))), \tag{5}$$

$\bar{s}(\cdot)$ adopts the same architecture as $s(\cdot)$ and can be used to extract the scale information for the inverse adaptive quantization layer. The network architecture of $s(\cdot)$ is shown in Table 1, where C is set to 80 in our implementation.

In [51], Choi *et al.* designed a variable bitrate scheme for image compression by changing the quantization step in the pre-defined range, and the features from different spatial locations share the same quantization step. However, the quantization step (i.e., scale information) in our approach is content adaptive, which is learned from the compressed features. More importantly, our proposed scheme does not change the existing baseline architecture, while Choi's approach uses conditional CNNs to replace all the standard CNN modules.

To evaluate the effectiveness of the proposed adaptive quantization layer, we provide the compression performance of our DVC_Pro method after using the adaptive quantization layers (named as DVC_Pro_AQ) in Fig. 10. It is noted that our DVC_Pro_AQ method achieves very similar compression

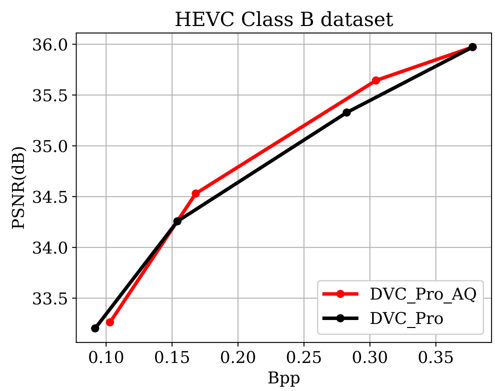
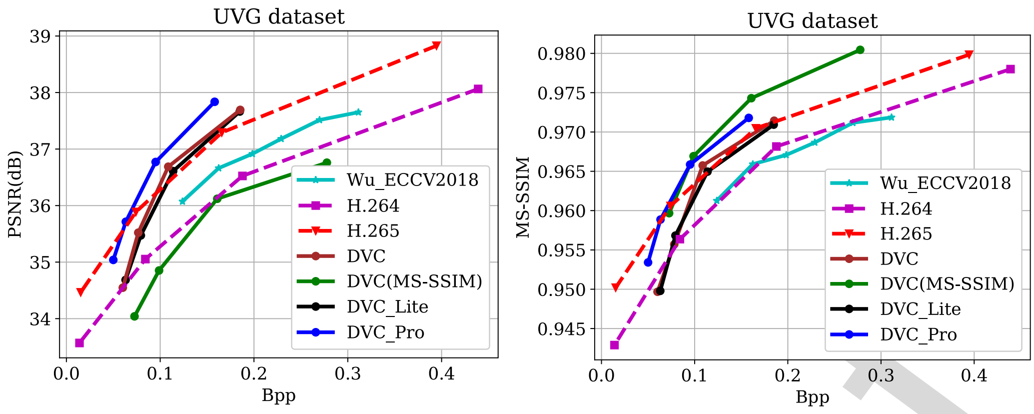


Fig. 10. Performance comparison between the separately trained DVC_Pro models and our newly proposed method DVC_Pro_AQ.



(a) Evaluation based on PSNR.

(b) Evaluation based on MS-SSIM.

Fig. 11. Performance comparison between our proposed method and the learning based video codec in Wu_ECCV2018 [23], H.264 [2], and H.265 [3] on the UVG dataset.

performance in most cases when compared with the separately trained DVC_Pro models at different bitrates. However, the number of parameters for the adaptive quantization layers in our DVC_Pro_AQ are only 2 percent of that from the full DVC_Pro model, which reduces the storage size for our models significantly.

7 Experiments

7.1 Experimental Setup

Datasets. We train the proposed video compression framework by using the Vimeo-90k dataset [24], which is recently built for evaluating different video processing tasks, such as video denoising and video super-resolution. It consists of 89,800 independent clips that are different from each other in content. The mini-batch size is set as 4 and the resolution of training images is 448×256.

To evaluate the performance of the proposed methods DVC/DVC_Lite/DVC_Pro, the UVG dataset [52], and the HEVC Standard Test Sequences (Class B, Class C, Class D, and Class E) [3] are used for evaluation. These datasets have diversified video content and resolutions, and thus are widely adopted to measure the performance of video compression algorithms in the literature.

Evaluation Method. To measure the distortion of the reconstructed frames, we use two evaluation metrics: PSNR and MS-SSIM [49]. MS-SSIM correlates better with the human perception of distortion than PSNR. To measure the number of bits for encoding the representations, we use bits per pixel(bpp) to represent the required bits for each pixel in the current frame.

In the video compression task, BDBR and BD-PSNR (BD-MSSSIM) [53] are widely used to evaluate the performance of different video compression systems. BDBR represents the average percentage of bit rate savings when compared with the baseline algorithm at the same PSNR (MS-SSIM). BD-PSNR (BD-MSSSIM) represents the performance gain(dB) when compared with the baseline algorithm at the same bit rate.

Deep Video Compression Models. There are four algorithm settings in our experiments. Specifically, the three approaches, which are optimized by using mean square error as the distortion metric,are denoted as DVC, DVC_Lite and DVC_Pro. Besides, we also report the results of another model that is optimized based on MS-SSIM [49], which is denoted as DVC (MS-SSIM). For each approach, we train 4 models with different trade-off parameter λ. For the MSE based models, the parameter λ is set to 256, 512, 1024 and 2048, respectively. Meanwhile, the corresponding λ values for the MS-SSIM based models are set to 8, 16, 32 and 64, respectively.

7.2 Experimental Results

In this section, both H.264 [2] and H.265 [3] are included for comparison. Furthermore, the recent learning based video compression system [23], denoted by Wu_ECCV2018, is also included for comparison. To generate the compressed frames by H.264 and H.265, we follow the setting in [23] and use FFmpeg with the *veryfast* mode.[1] For fair comparison, both the proposed approach and the baseline methods in Figs. 11, 12, and 13 use the same GoP size. Specifically, the GoP size for the UVG dataset and the HEVC dataset are 12 and 10, respectively.

PSNR Evaluation. Figs. 11a and 12 show the PSNR based rate-distortion performance on the UVG dataset and the HEVC standard test sequences (Class B, Class C, Class D, Class E), respectively. From Fig. 11a, it is noticed that our MSE based models DVC/DVC_Lite/DVC_Pro outperform the recent video compression work [23] by a large margin. Specifically, the proposed DVC model achieves about 0.6dB gain at the same bpp level on the UVG dataset. It should be mentioned that our method only uses one previous reference frame, while the work by Wu *et al.* [23] utilizes bidirectional frame prediction and requires two neighboring frames. In other words, our new framework for P-frame compression surpasses the B-frame compression method in [23]. A possible explanation is that we jointly optimize all

1. H.264: *ffmpeg -pix_fmt yuv420p -s WxH -r FR -i Video.yuv -vframes N -c:v libx264 -preset veryfast -tune zerolatency -crf Q -g GOP-bf 2 -b_strategy 0 -sc_threshold 0 output.mkv*

H.265: *ffmpeg -pix_fmt yuv420p -s WxH -r FR -i Video.yuv -vframes N -c:v libx265 -preset veryfast -tune zerolatency -x265-params "crf=Q:keyint=GOP" output.mkv*

FR, N, Q, GOP represent the frame rate, the number of encoded frames, the quality and GOP size, respectively. N is set to 100 for the HEVC datasets. GOP is set as 10 for the HEVC dataset and 12 for the UVG dataset.

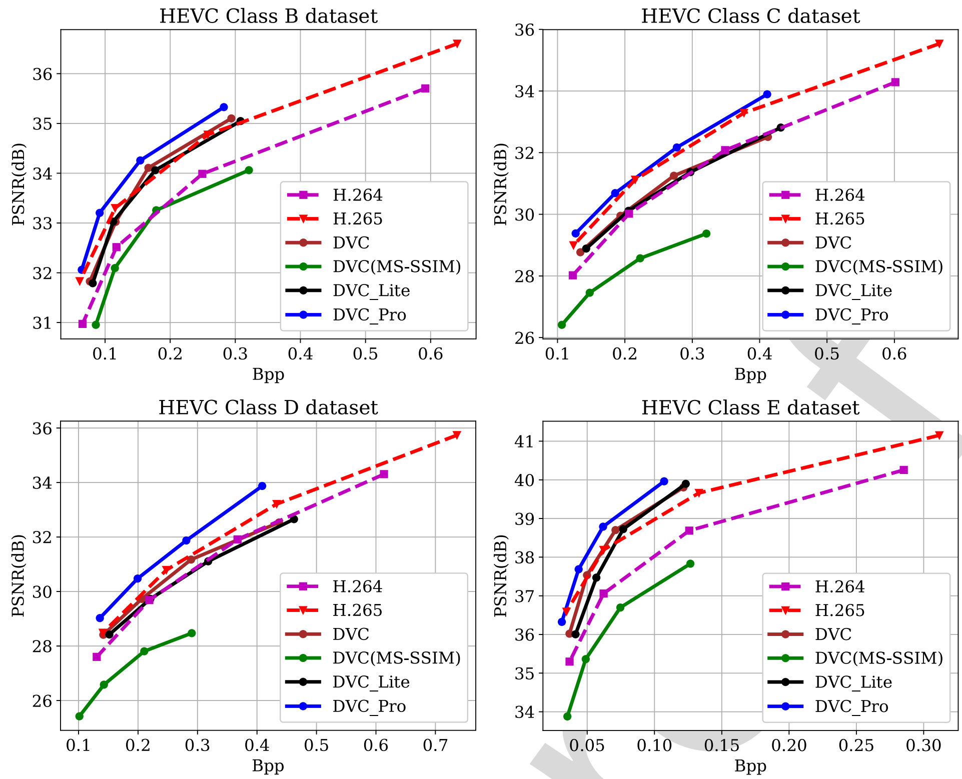


Fig. 12. Performance(PSNR) comparison between our proposed method and H.264 [2], H.265 [3] on the HEVC dataset.

the components in our framework while the motion information in [23] is not optimized in an end-to-end fashion.

On most datasets, our MSE based model outperforms the H.264 standard when measured by PSNR. It can also be observed that the performance of DVC_Lite is similar to DVC, while DVC_Pro outperforms DVC, especially for the HEVC Class C dataset in Fig. 12. More importantly, the DVC_Pro model even achieves comparable compression performance with H.265 in terms of PSNR, which demonstrates the potential of the learning based video compression approach.

In Table 2, we provide the BDBR and BD-PSNR results of H.265 and our proposed methods DVC/DVC_Lite/DVC_Pro when compared with H.264. Specifically, our proposed DVC model saves 19.22 percent bit rate, while H.265 saves 25.06 percent bit rate. However, the advanced model DVC_Pro saves up to 34.57 percent bit rate, which outperforms H.265.

We also observe that the DVC_Lite model achieves similar compression performance when compared with the DVC model. However, DVC_Lite only requires 27 percent of the total number of parameters and reduces 76 percent FLOPs. More analysis of the computational complexity is provided in Section 7.4.

Evaluations under the Default Setting of H.265. The previous learning based video compression methods [23], [25], [38] use the fixed GoP size when evaluating the performance between different codecs. To provide a more comprehensive evaluation, we also compare our method with the default setting of x265,[2] where the variable large GoP size is utilized. All the video frames in HEVC Class B and Class C are used for performance evaluation. From the experimental results in Fig. 14, it is observed that the proposed method can achieve very competitive results on the HEVC Class B dataset at high bitrates. Considering that our approach does not exploit sophisticated rate control techniques, multiple reference frames, etc, our method still achieves promising results.

MS-SSIM Evaluation. In Figs. 11b and 13, the MS-SSIM based rate-distortion performances are provided. Although DVC, DVC_Lite and DVC_Pro are optimized by minimizing the MSE, these methods have also achieved promising MS-SSIM performance. For example, the rate-distortion curves in Fig. 13 show that our MSE based models achieve comparable or better compression performance than H.265 in terms of MS-SSIM. It demonstrates that our framework can generate reconstructed frames with better perceptual quality.

For the model optimized based on MS-SSIM, the proposed DVC(MS-SSIM) method outperforms the H.265/H.264 codecs by a large margin when measured by MS-SSIM. For example, the DVC(MS-SSIM) method achieves more than 0.005 gain in terms of MS-SSIM when compared with H.265 on the HEVC Class C dataset. Besides, when measured by PSNR, it is also not surprising that the performance of DVC(MS-SSIM) method in Fig. 11a decreases significantly. One possible explanation is that the network based on the MS-SSIM criterion prefers to preserve the structure of the reconstructed image instead of preserving the original pixel intensity, which leads to performance drop in terms of PSNR.

We also provide the BDBR and BD-MSSSIM results in Table 3. It is observed that our proposed DVC method can save more than 29 percent bit rate over all sequences, while

2. Command line for FFmpeg: *ffmpeg -pix_fmt yuv420p -s WxH -r 50 -i video.yuv -c:v libx265 -tune zerolatency -x265-params "qp=Q" output.mkv* Q is the quantization parameter. W and H are the height and width of the yuv video.

TABLE 2
BDBR(%) and BD-PSNR(dB) Performances of H.265 and Our DVC/DVC_Lite/DVC_Pro Methods When Compared With H.264 on the HEVC Standard Test Sequences in Terms of PSNR

Sequences		H.265		DVC		DVC_Lite		DVC_Pro	
		BDBR	BD-PSNR	BDBR	BD-PSNR	BDBR	BD-PSNR	BDBR	BD-PSNR
B	BasketballDrive	-44.37	1.15	-23.17	0.59	-24.91	0.61	**-55.09**	**1.53**
	BQTerrace	-28.99	0.68	-25.12	0.54	-8.11	0.16	**-36.62**	**0.88**
	Cactus	-30.15	0.68	-39.53	0.94	-31.60	0.78	**-46.50**	**1.18**
	Kimono	-38.81	1.19	-40.70	1.23	-39.05	1.28	**-52.31**	**2.09**
	ParkScene	-16.35	0.45	-25.20	0.77	-20.29	0.63	**-34.20**	**1.08**
	Average	-31.73	0.83	-30.75	0.81	-24.79	0.69	**-44.94**	**1.35**
C	BasketballDrill	-35.08	1.69	-24.47	1.05	-22.22	0.96	**-35.48**	**1.75**
	BQMall	-19.70	0.84	26.13	-0.72	24.74	-0.75	**-20.77**	**0.93**
	PartyScene	-13.41	0.60	-9.14	0.29	-5.24	0.16	**-18.14**	**0.81**
	RaceHorses	-17.28	0.69	-8.06	0.19	-3.49	0.12	**-27.79**	**1.13**
	Average	-21.37	0.96	-3.88	0.20	-1.55	0.12	**-25.54**	**1.15**
D	BlowingBubbles	-12.51	0.50	-17.79	0.62	-10.94	0.36	**-26.19**	**1.09**
	BasketballPass	-19.26	0.99	-0.39	-0.01	7.28	-0.31	**-20.60**	**1.02**
	BQSquare	-3.49	0.14	-1.60	0.01	5.40	-0.25	**-17.96**	**0.82**
	RaceHorses	-14.77	0.68	-18.95	0.72	-7.63	0.34	**-30.17**	**1.51**
	Average	-12.51	0.58	-9.68	0.34	-1.47	0.04	**-23.73**	**1.11**
E	Vidyo1	-37.12	1.11	-36.05	1.20	-29.10	0.88	**-44.09**	**1.67**
	Vidyo3	-34.99	1.23	-32.58	1.25	-25.37	0.86	**-41.02**	**1.72**
	Vidyo4	-34.71	1.05	-30.84	1.03	-26.51	0.86	**-46.25**	**1.73**
Average		-35.61	1.13	-33.16	1.16	-26.99	0.87	**-43.79**	**1.72**
Average Over All Sequences		-25.06	0.85	-19.22	0.61	-13.56	0.42	**-34.57**	**1.31**

The best result in each row is highlighted in bold.

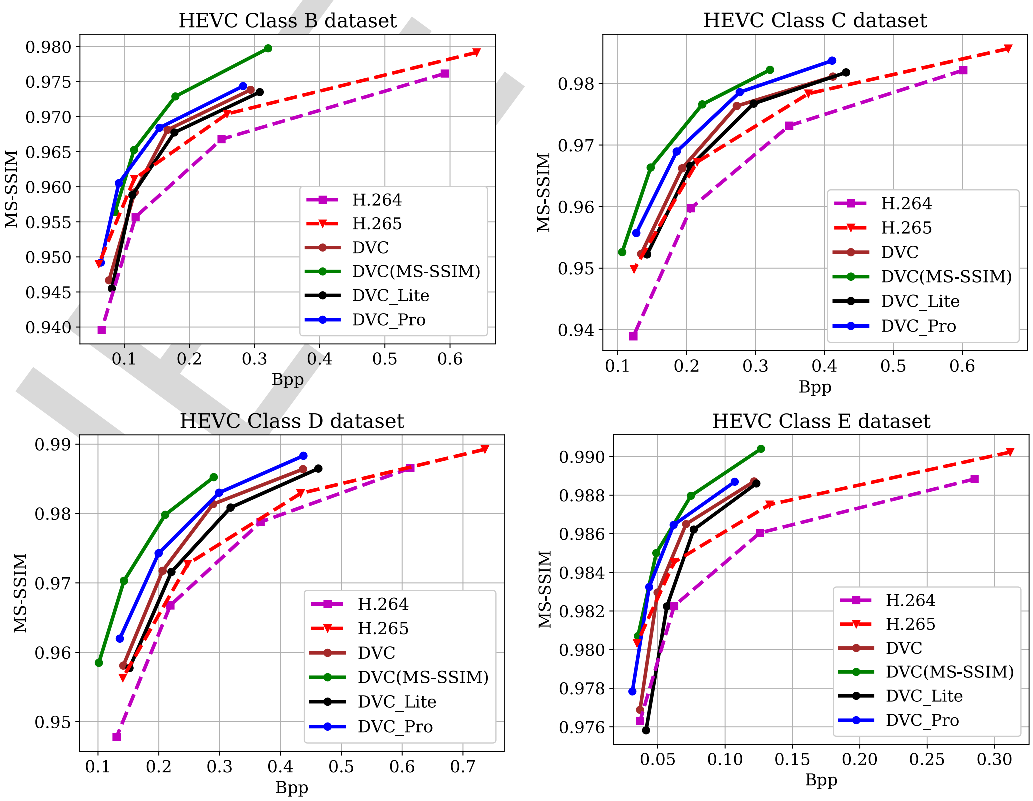


Fig. 13. Performance(MS-SSIM) comparison between our proposed method and H.264 [2], H.265 [3] on the HEVC dataset.

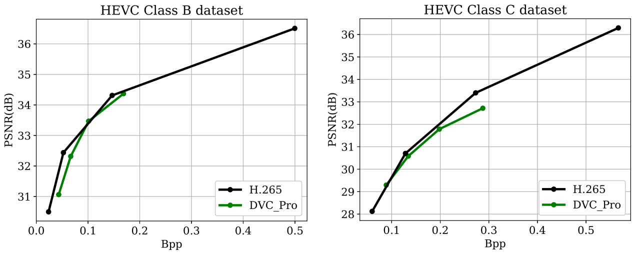


Fig. 14. Results of our DVC_Pro method and H.265 (with the default setting in FFmpeg) on the HEVC dataset.

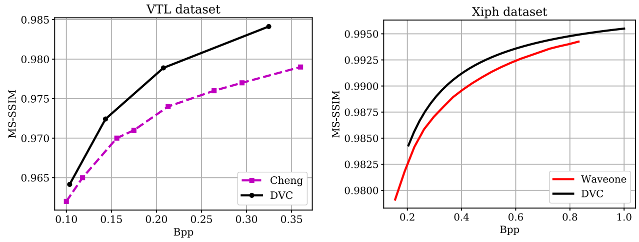


Fig. 15. Evaluation results with the state-of-the-art methods Cheng [38] and Waveone [54] on the VTL and Xiph datasets.

H.265 only saves 21.73 percent bitrate. Besides, the MS-SSIM based method DVC(MS-SSIM) saves up to 45.88 percent bit rate when compared with H.264. Based on the results from Tables 2 and 3, it clearly demonstrates that our proposed method outperforms H.264 in terms of PSNR and MS-SSIM.

To further demonstrate the effectiveness of the proposed approach, we also compare our DVC method with the latest learning based approaches [38], [54] on the VTL [55] and Xiph [56] datasets. The experimental results are shown in Fig. 15 and it is observed that our approach outperforms these approaches by a large margin. For example, when compared with Cheng's approach [38] at 0.25bpp, our DVC model achieves 0.005 improvements in terms of MS-SSIM and reduces 25 percent bitrate when evaluated based on BDBR.

Qualitative Comparison. In Fig. 16, the reconstructed frames from different video compression algorithms are provided. Specifically, when compared with H.264/H.265, our DVC method generates high-quality reconstructed frames at the same bpp level. For example, our DVC method can generate a clear contour of the digital number in the top row of Fig. 16, while other methods generate more blurry contour.

7.3 Ablation Study and Model Analysis

In this section, we provide the ablation study of our proposed DVC method.

Motion Estimation. In our proposed method, we exploit the advantage of the end-to-end training strategy and optimize the motion estimation module within the whole network. Therefore, based on rate-distortion optimization, the optical flow map in our system is expected to be more compressible, leading to more accurate warped frames. To demonstrate the effectiveness, we perform an experiment by fixing the parameters of the initialized motion estimation module in the whole training stage. In this case, the motion estimation module is pre-trained only for estimating accurate optical flow, but not for optimal rate-distortion. The experimental result in Fig. 17 shows that

TABLE 3
BDBR(%) and BD-MSSSIM(dB) Performances of H.265 and our DVC/DVC_Lite/DVC_Pro Methods When Compared With H.264 on the HEVC Standard Test Sequences in Terms of MS-SSIM

Sequences		H.265		DVC		DVC_Lite		DVC_Pro		DVC(MS-SSIM)	
		BDBR	BD-MSSSIM	BDBR	BD-MSSSIM	BDBR	BD-MSSSIM	BDBR	BD-MSSSIM	BDBR	BD-MSSSIM
B	BasketballDrive	-39.80	0.87	-22.21	0.51	-23.84	0.50	**-52.55**	**1.40**	-48.99	1.16
	BQTerrace	-25.96	0.50	-19.52	0.36	-2.00	-0.11	-26.39	0.48	**-43.21**	**0.99**
	Cactus	-26.93	0.47	-41.71	0.86	-35.36	0.72	-47.84	1.01	**-53.69**	**1.22**
	Kimono	-35.31	0.97	-33.00	0.92	-31.67	0.92	-46.11	1.69	**-51.27**	**1.53**
	ParkScene	-13.54	0.29	-29.02	0.77	-24.47	0.64	-35.93	1.03	**-44.76**	**1.29**
	Average	-28.31	0.62	-29.09	0.68	-23.46	0.53	-41.76	1.12	**-48.39**	**1.24**
C	BasketballDrill	-34.04	1.41	-27.18	1.18	-24.07	1.03	-34.64	1.77	**-43.54**	**2.21**
	BQMall	-17.57	0.60	-18.85	0.67	-15.51	0.54	-32.07	1.41	**-40.17**	**1.75**
	PartyScene	-13.36	0.53	-37.18	1.61	-32.39	1.43	-37.41	1.90	**-41.09**	**2.08**
	RaceHorses	-17.01	0.57	-29.24	1.05	-22.98	0.84	-37.97	1.59	**-43.28**	**1.89**
	Average	-20.50	0.78	-28.11	1.13	-23.74	0.96	-35.52	1.67	**-42.02**	**1.98**
D	BlowingBubbles	-10.28	0.35	-35.44	1.53	-29.79	1.25	-38.89	1.70	**-46.53**	**2.20**
	BasketballPass	-17.98	0.85	-20.53	1.01	-14.54	0.65	-28.44	1.68	**-43.28**	**2.60**
	BQSquare	5.90	-0.19	-23.67	0.84	-14.59	0.52	-23.27	1.05	**-28.14**	**1.86**
	RaceHorses	-13.23	0.56	-29.79	1.30	-20.14	0.89	-34.93	1.87	**-46.32**	**2.39**
	Average	-8.89	0.39	-27.36	1.17	-19.76	0.83	-31.38	1.58	**-41.07**	**2.26**
E	Vidyo1	-31.67	0.55	-36.80	0.72	-21.39	0.33	-33.97	1.00	**-56.73**	**1.22**
	Vidyo3	-29.48	0.65	-40.09	1.02	-19.46	0.35	-39.83	1.29	**-53.87**	**1.51**
	Vidyo4	-27.41	0.61	-24.84	0.66	-14.51	0.34	-37.04	1.15	**-49.19**	**1.33**
	Average	-29.52	0.61	-33.91	0.80	-18.45	0.34	-36.95	1.15	**-53.26**	**1.35**
Average Over All Sequences		-21.73	0.60	-29.32	0.94	-21.67	0.68	-36.70	1.38	**-45.88**	**1.70**

The best result in each row is highlighted ***in bold****.*

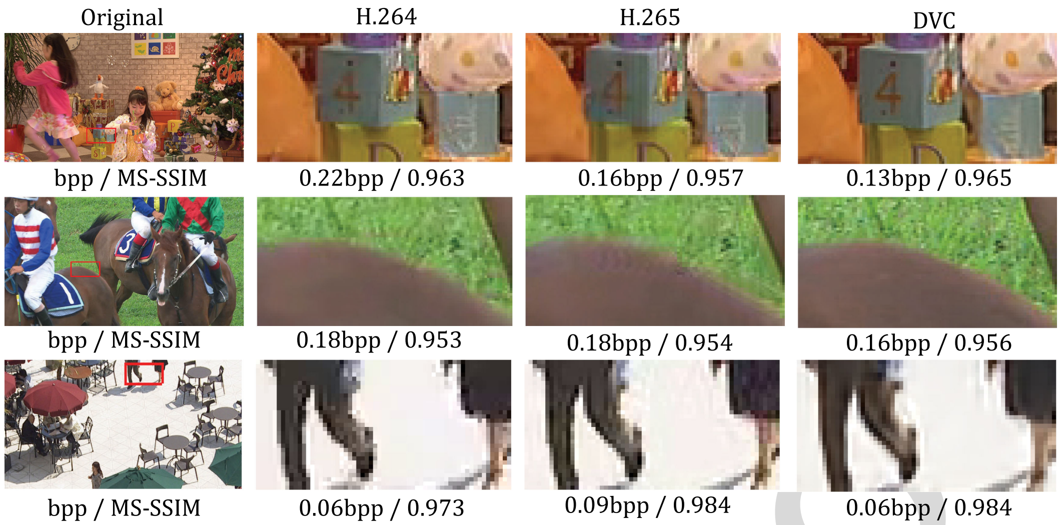


Fig. 16. Qualitative comparison. The reconstructed frames are from H.264, H.265, and our DVC method. Our method either achieves better visual quality or uses fewer bits.

our approach with the joint training strategy can improve the performance significantly when compared with the approach by fixing the motion estimation module, which is denoted by *W/O Joint Training* in Fig. 17 (see the blue curve).

We also report the average bit costs for encoding the optical flow maps and the corresponding PSNR of the warped frame in Table 4. It is observed that we can obtain a higher quality warped frame with fewer bits cost by using the optical flow map based on the joint training strategy. Specifically, when the motion estimation module is fixed during the training stage, it needs 0.044bpp to encode the generated optical flow map and the corresponding PSNR of the warped frame is 27.33db. In contrast, we need 0.029bpp to encode the optical flow map in our proposed method, and the PSNR of the warped frame is 28.17dB, which is higher. Therefore, the joint learning strategy not only saves the number of bits required for encoding motion information but also improves the warped image quality. These experimental results clearly show that video compression performance can be improved by putting the motion estimation module into rate-distortion optimization.

Motion Estimation Based on Downsampled Frames. In Section 6.1, we propose an efficient optical flow estimation network based on the downsampled frames to reduce the computational complexity while achieving high-quality motion estimation results. Our scheme upsamples the estimated optical flow maps based on the context information from high resolution frames. To demonstrate the effectiveness of such context information, we also perform a new experiment by removing the context information. Specifically, the optical flow maps based on downsampled frames will be upsampled by bilinear interpolation and compressed by the following motion compression network. Although the encoding speed can be improved by 15 percent after removing such context information, the coding performance drops more than 0.4 dB. Therefore, it is necessary to use context information from high resolution frames when performing the upsampling operation.

Motion Compensation. In this work, the motion compensation network is utilized to refine the warped frames. Since the motion estimation module may generate unreliable optical flow map, it is necessary to refine the warped frames. To evaluate the effectiveness of this module, we perform another experiment by removing the motion compensation network in the proposed system. Specifically, we use the warped frame $\tilde{x}_t$ in Eq. (1) as the predicted frame $\bar{x}_t$ without using the CNN network for refinement. Experimental results of this alternative approach, which is denoted by *W/O MC* (see the green curve in Fig. 17), show that the PSNR without the motion compensation network drops by 1.0 dB at the same bpp level.

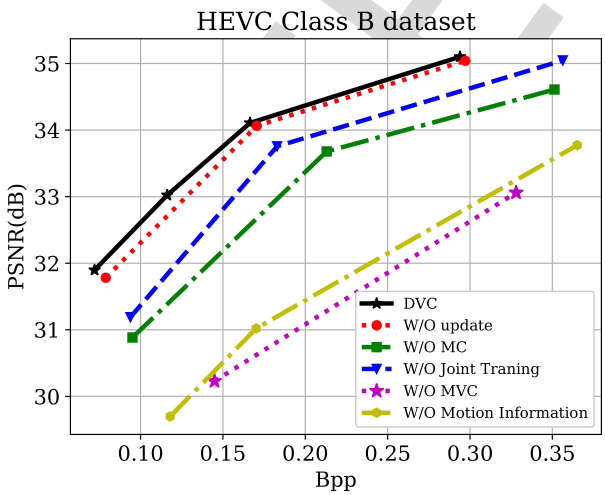


Fig. 17. Ablation study. We report the compression performance in the following settings. 1. The strategy of buffering the previous frame is not adopted(W/O update). 2. Motion compensation network is removed (W/O MC). 3. Motion estimation module is not jointly optimized (W/O Joint Training). 4. Motion compression network is removed (W/O MVC). 5. Without relying on motion information (W/O Motion Information).

TABLE 4
The Bit Cost for Encoding Optical Flow Maps and the Corresponding PSNR of the Warped Frame

W/O Joint Training		W/O MVC		DVC	
Bpp	PSNR	Bpp	PSNR	Bpp	PSNR
0.044	27.33	0.200	24.43	0.029	28.17

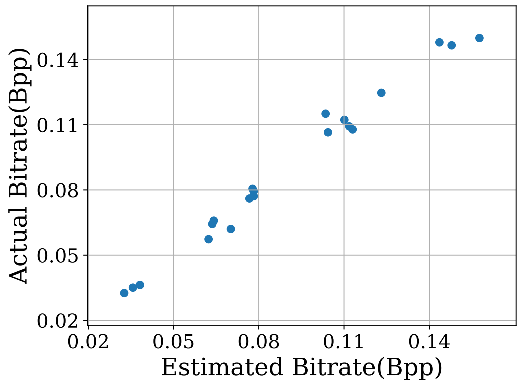


Fig. 18. Comparison between the actual bitrate through arithmetic entropy coding and the estimated bit rate in our network at different bpps.

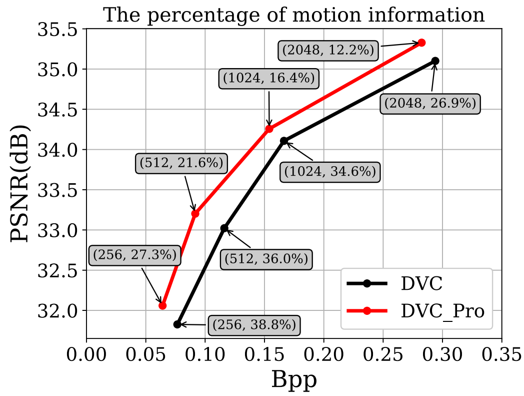


Fig. 19. Percentages of bits used to encode motion information at different bpps. (λ, p) represent the trade-off parameter in Eq. (2) and the percentage of bits used to encode motion information, respectively.

Updating Strategy. In the training stage for our proposed method, the previous reconstructed frame is required for encoding the current frames. To address this issue, we use an online buffer in Section 5 to store previously reconstructed frames in the training stage when encoding the current frame x_t. To demonstrate the effectiveness of the proposed scheme, we also report the compression performance when the previous reconstructed frame $\hat{x}_{t-1}$ is directly replaced by the previous original frame x_{t-1} in the training stage. This result of this alternative approach, which is denoted by *W/O update* (see the red curve), is shown in Fig. 17. It is observed that the buffering strategy can improve the performance at the same bpp level. One explanation is that updating strategy can generate accurate reference frames, which benefits the training procedure.

MV Encoder and Decoder Network. In our proposed framework, we design a CNN model to compress the optical flow and encode the corresponding motion representations. It is also feasible to directly quantize the raw optical flow values and encode them without using any CNN. Specifically, we perform a new experiment by removing the MV encoder and decoder network. The experimental result in Fig. 17 shows that the PSNR of the alternative approach, which is denoted by *W/O MVC* (see the magenta curve), drops by more than 2 dB after removing the motion compression network. Besides, the bit cost for encoding the optical flow map in this setting and the corresponding PSNR of the warped frame are also provided in Table 4 (denoted by *W/O_MVC*). It is noticed that it requires much more bits (0.200 Bpp) to directly encode raw optical flow values and the corresponding PSNR(24.43 dB) is much worse than our proposed method(28.17 dB). Therefore, motion compression is crucial when optical flow is used in the learning based video codec.

Motion Information. To demonstrate the effectiveness of motion information for video compression, we also investigate the alternative approach, which only retains the residual encoder and decoder network. As shown in Fig. 17, when treating each frame independently without using any motion estimation approach (see the yellow curve denoted by *W/O Motion Information*, the PSNR performance drops more than 2dB when compared with our baseline method.

Bit Rate Analysis. In this paper, we use a probability estimation network in [8] to estimate the bit rate for encoding motion and residual information. To verify the reliability, we compare the estimated bit rate and the actual bit rate by using arithmetic coding in Fig. 18. It is observed that the estimated bit rate is close to the actual bit rate, which leads to accurate rate-distortion optimization. Furthermore, we investigate the two parts in the rate-distortion function in Eq. (2). In Fig. 19, we provide the λ value and the percentage of motion information at each point. When the parameter λ in our objective function $\lambda D + R$ becomes larger, the whole bpp also becomes larger while the corresponding percentage of bits used for encoding motion information drops. In other words, our video compression framework will use more bits to encode the residual at a high bit rate.

7.4 Computational Complexity Analysis

The computational complexity and compression efficiency are the two most important metrics for practical video compression systems. In this section, we provide an in-depth analysis of computational complexity for the proposed methods DVC/DVC_Lite/DVC_Pro.

Encoding Speed. To compare the computational complexity of different video compression systems, we perform several experiments by using the server with Intel Xeon E5-2640 v4 CPU and a single GTX 1080Ti GPU. Specifically, we include the two official reference software JM [57] and HM [58], two practical commercial software x264 [59] and x265 [60], the learning based methods from [23] and our proposed methods DVC/DVC_Lite/DVC_Pro. The experimental results are provided in Fig. 20. For the video sequences with the resolution of 1920x1080, the encoding speeds of JM and HM are 0.14fps and 0.02fps, respectively. In contrast, the encoding speed of our DVC_Light model is 3.33fps, which is *23.7* times faster than JM and *166* times faster than HM. Experimental results in Fig. 20 also demonstrate that our DVC_Lite is *2* times faster than the DVC method in [25]. Considering that the DVC_Lite method achieves similar coding performance as the DVC method, it demonstrates the effectiveness of our newly proposed motion estimation and motion compression networks. The encoding speed of our DVC_Pro is 1.35fps.

Besides, the practical video codecs x264 and x265 have different coding settings to balance the coding efficiency and encoding speed. For example, when the coding efficiency is given higher priority for video codecs, the

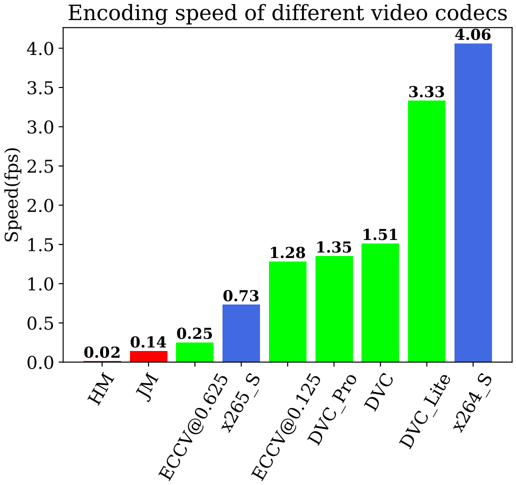


Fig. 20. Encoding speed of different video codecs.

corresponding speeds are 4.06fps(x264_S) and 0.73fps (x265_S), which are similar to our DVC/DVC_Lite model. For the *faster* setting in x264 and x265, the encoding speed can be up to 109fps and 16fps, respectively.

It should be mentioned that both x264 and x265 are developed based on the highly parallel framework and use the assembly optimization techniques, which lead to the state-of-the-art encoding speed. Recently, a lot of deep model acceleration techniques, such as model pruning or model quantization, have been widely used to improve the inference time and reduce model size. Therefore, it is possible to further improve the speed of our method by using the latest techniques.

We also provide the encoding speed of Wu's framework [23] in Fig. 20. Since the progressive coding scheme is utilized in [23], the coding speed changes for different target bitrate, i.e., different iterations. For the high bitrate, denoted as ECCV@0.625, the corresponding coding speed is 0.25fps, while our DVC_Lite is *13.2* times faster. For the low bitrate, denoted as ECCV@0.125, the encoding speed is 1.28fps, and DVC_Lite is about *2.6* times faster in speed.

In addition, another advantage of our proposed model is the *complexity invariance*. Specifically, for the given video sequence with specific resolution, the encoding speed of

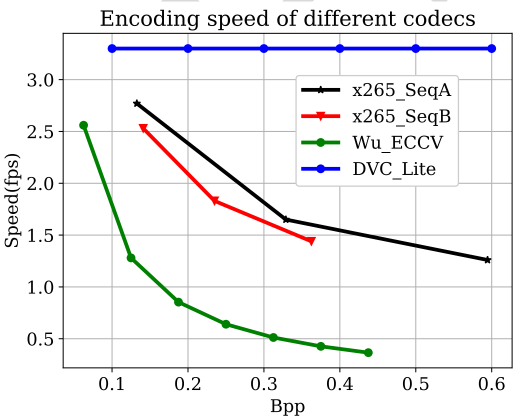


Fig. 21. Encoding speed of different video codecs for different video contents and bpps. x265_SeqA and x265_SeqB represent the speed of x265 codec (the *slower* setting) for different video sequences at various bitrates.

TABLE 5
The Trainable Parameters and FLOPs for Different Models

Methods	Parameters	FLOPs
DVC	10.1M	154.7G
DVC_Lite	2.7M	36.9G
ImageCodec	11.8M	29.0G

ImageCodec represents the learning based image codec in [8] (the number of channels in the bottleneck layer is set to 320).

our model keeps constant irrespective of the target bitrate or the video content. However, as shown in Fig. 21, the encoding speed of the learning based method [23] and the traditional video codecs vary a lot for different bitrates. Due to the existing mode decision scheme in the traditional video codecs, the corresponding encoding speed also varies for different video content even at the same bitrate.

Model Complexity. In Table 5, we provide the parameters and FLOPs of our proposed method. Specifically, for the video sequence with the resolution of 384x192, the corresponding parameters and FLOPs of our DVC method are 10.1M and 154.7GFLOPs. In comparison with DVC, the parameters and FLOPs of the newly proposed DVC_Lite are 2.7M (*73 percent reduction*) and 36.9GFLOPs (*76 percent reduction*), respectively. For the DVC_Pro model, the parameters and FLOPs are 29.4M and 294.6GFLOPs. Furthermore, we also provide the trainable parameters and FLOPs for the learning based image codec [8]. It can be observed that the FLOPs of DVC_Lite are comparable with the image codec in [8] (36.9GFLOPs vs. 29GFLOPs). However, our model is much smaller (2.7M vs. 11.8M). The results demonstrate the efficiency of our proposed video compression codec.

When comparing DVC_Lite with DVC, the reduction ratios of each sub-network are illustrated in Fig. 22. Specifically, the proposed motion estimation scheme based on the downsampled frames can reduce up to 92 percent FLOPs when compared with the original motion estimation network in [25]. Instead of using GDN [5] and building a large-capacity network to compress the motion information, we find that a lightweight and efficient network is good enough to compress the motion information, which leads to 74 percent reduction in the model size. Besides, we also reduce the channel number of the motion compensation network and the residual compression network to decrease the computational complexity. For example, we reduce the channel number of the residual compression network from 128 to 64 and obtain 83 percent reduction in FLOPs.

8 Conclusion

In this paper, we have proposed the fully end-to-end deep learning framework for video compression. Our framework inherits the advantages of both classic predictive coding scheme in the traditional video compression standards and the powerful non-linear representation ability from DNNs. Experimental results show that our approach outperforms the widely used H.264 video compression standard and the recent learning based video compression system. The work provides a promising framework for applying deep neural networks for video compression. Based on the proposed framework, new state-of-the-art methods for optical flow

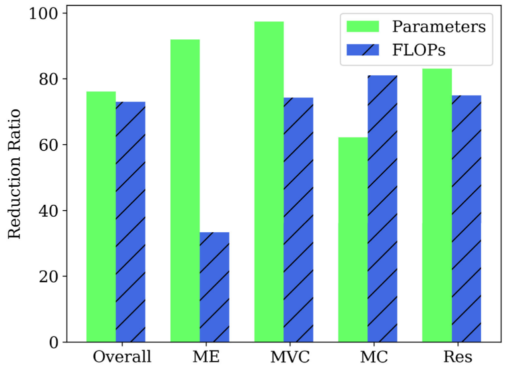


Fig. 22. Reduction ratios of parameters and FLOPs when comparing the corresponding sub-networks in the DVC_Lite and DVC. 'Overall' represents the ratios when comparing the full model of DVC_Lite with DVC. And 'ME' represents the Motion Estimation Net in Fig. 1b. 'MVC' represents the MV Encoder Net and MV Decoder Net in Fig. 1b. 'MC' represents the Motion Compensation Net in Fig. 1b. 'Res' represents the Residual Encoder Net and Residual Decoder Net in Fig. 1b.

estimation, image compression, bi-directional prediction, and rate control can be readily plugged into this framework.

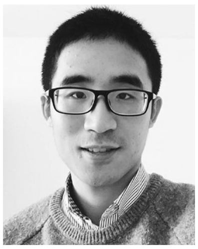
Acknowledgments

This work was supported in part by National Natural Science Foundation of China (61771306), Natural Science Foundation of Shanghai(18ZR1418100), 111 plan (B07022), Shanghai Key Laboratory of Digital Media Processing and Transmissions(STCSM 18DZ2270700). This work was also supported by the Australian Research Council (ARC) Future Fellowship under Grant FT180100116.

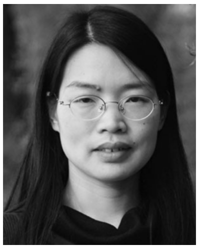
References

[1] C. V. networking Index, "Forecast and methodology, 2016–2021, White Paper," San Jose, CA, USA, vol. 1, 2016.
[2] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H. 264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.
[3] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.
[4] G. Toderici *et al.*, "Variable rate image compression with recurrent neural networks," in *Proc. 4th Int. Conf. Learn. Representations*, 2016.
[5] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," in *Proc. 5th Int. Conf. Learn. Representations*, 2017.
[6] G. Toderici *et al.*, "Full resolution image compression with recurrent neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5435–5443.
[7] E. Agustsson *et al.*, "Soft-to-hard vector quantization for end-to-end learning compressible representations," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1141–1151.
[8] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *Proc. 6th Int. Conf. Learn. Representations*, 2018.
[9] N. Johnston *et al.*, "Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4385–4393.
[10] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," in *Proc. 5th Int. Conf. Learn. Representations*, 2017.
[11] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang, "Learning convolutional networks for content-weighted image compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3214–3223.
[12] O. Rippel and L. Bourdev, "Real-time adaptive image compression," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 2922–2930.
[13] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, and L. Van Gool, "Generative adversarial networks for extreme learned image compression," In *Pro. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 221–231.
[14] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consum. Electron.*, vol. 38, no. 1, pp. xviii–xxxiv, Feb. 1992.
[15] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG 2000 still image compression standard," *IEEE Signal Process. Magazine*, vol. 18, no. 5, pp. 36–58, Sep. 2001.
[16] F. bellard, "BPG image format," Accessed: Oct. 30, 2018, [Online]. Available: http://bellard.org/bpg/
[17] D. Liu, Y. Li, J. Lin, H. Li, and F. Wu, "Deep learning-based video coding: A review and a case study," *ACM Comput. Surveys*, vol. 53, no. 1, pp. 1–35.
[18] T. Chen, H. Liu, Q. Shen, T. Yue, X. Cao, and Z. Ma, "Deepcoder: A deep neural network based video compression," in *Proc. IEEE Vis. Commun. Image Process.*, 2017, pp. 1–4.
[19] Z. Liu, X. Yu, Y. Gao, S. Chen, X. Ji, and D. Wang, "CU partition mode decision for HEVC hardwired intra encoder using convolution neural network," *IEEE Trans. Image Process.*, vol. 25, no. 11, pp. 5088–5103, Nov. 2016.
[20] R. Song, D. Liu, H. Li, and F. Wu, "Neural network-based arithmetic coding of intra prediction modes in VEVC," in *Proc. IEEE Vis. Commun. Image Process. Conf.*, 2017, pp. 1–4.
[21] G. Lu, W. Ouyang, D. Xu, X. Zhang, Z. Gao, and M.-T. Sun, "Deep kalman filtering network for video compression artifact reduction," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 591–608.
[22] R. Yang, M. Xu, Z. Wang, and T. Li, "Multi-frame quality enhancement for compressed video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6664–6673.
[23] C.-Y. Wu, N. Singhal, and P. Krahenbuhl, "Video compression through image interpolation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 425–440.
[24] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video enhancement with task-oriented flow," *Int. J. Comput. Vis.*, vol. 127, no. 8, pp. 1106–1125, 2019.
[25] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao, "DVC: An end-to-end deep video compression framework," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 11 006–11 015.
[26] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool, "Conditional probability models for deep image compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, Art. no. 3.
[27] D. Minnen, J. Ballé, and G. D. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," in *Proc. Advances Neural Inf. Process. Syst.*, 2018, pp. 10 771–10 780.
[28] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, and L. V. Gool, "Generative adversarial networks for extreme learned image compression," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 221–231.
[29] Y. Patel, S. Appalaraju, and R. Manmatha, "Deep perceptual compression," 2019, *arXiv: 1907.08310*.
[30] J. Lee, S. Cho, and S.-K. Beack, "Context-adaptive entropy model for end-to-end optimized image compression," in *Proc. 7th Int. Conf. Learn. Representations*, 2019.
[31] Y. Blau and T. Michaeli, "Rethinking lossy compression: The rate-distortion-perception tradeoff," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 675–685.
[32] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.
[33] D. Minnen *et al.*, "Spatially adaptive image compression using a tiled deep network," in *Proc. IEEE Int. Conf. Image Process.*, 2017, pp. 2796–2800.
[34] M. H. Baig, V. Koltun, and L. Torresani, "Learning to inpaint for image compression," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1246–1255.
[35] Y. Patel, S. Appalaraju, and R. Manmatha, "Human perceptual evaluations for image compression," 2019, *arXiv: 1908.04187*.
[36] Z. Chen, T. He, X. Jin, and F. Wu, "Learning for video compression," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 30, no. 2, pp. 566–576, 2020.
[37] Y.-H. Tsai, M.-Y. Liu, D. Sun, M.-H. Yang, and J. Kautz, "Learning binary residual representations for domain-specific video streaming," in *Proc. 32nd AAAI Conf. Artif. Intell.*, pp. 7363–7370, 2018.
[38] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Learning image and video compression through spatial-temporal energy compaction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10 071–10 080.

[39] "A new image format for the Web," Accessed: Oct. 30, 2018, [Online]. Available: https://developers.google.com/speed/webp/
[40] Z. Chen, J. Xu, Y. He, and J. Zheng, "Fast integer-pel and fractional-pel motion estimation for H. 264/AVC," *J. Vis. Commun. Image Representation*, vol. 17, no. 2, pp. 264–290, 2006.
[41] C.-Y. Chen, S.-Y. Chien, Y.-W. Huang, T.-C. Chen, T.-C. Wang, and L.-G. Chen, "Analysis and architecture design of variable block-size motion estimation for H. 264/AVC," *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 53, no. 3, pp. 578–593, Mar. 2006.
[42] A. Barjatya, "Block matching algorithms for motion estimation," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, Apr. 2004.
[43] A. Dosovitskiy *et al.*, "Flownet: Learning optical flow with convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2758–2766.
[44] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proc. EEE Conf. Comput. Vis. Pattern Recognit.*, 2017, Art. no. 2.
[45] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-net: CNNs for optical flow using pyramid, warping, and cost volume," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8934–8943.
[46] T.-W. Hui, X. Tang, and C. Change Loy, "Liteflownet: A lightweight convolutional neural network for optical flow estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8981–8989.
[47] T.-W. Hui, X. Tang, and C. C. Loy, "A lightweight optical flow CNN–revisiting data fidelity and regularization," in *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019.
[48] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Proc. Advances Neural Inf. Process. Syst.*, 2015, pp. 2017–2025.
[49] Z. Wang, E. Simoncelli, and A. Bovik, "Multi-scale structural similarity for image quality assessment," in *Proc. 37th Asilomar Conf. Signals Syst. Comput.*, 2003, pp. 1398–1402.
[50] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, in *Proc. 3rd Int. Conf. Learn. Representations*, 2015.
[51] Y. Choi, M. El-Khamy, and J. Lee, "Variable rate deep image compression with a conditional autoencoder," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 3146–3154.
[52] "Ultra video group test sequences," Accessed: Oct. 30, 2018, [Online]. Available: http://ultravideo.cs.tut.fi
[53] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," in *VCEG Meeting (ITU-T SG16 Q. 6)*, pp. 2–4, 2001.
[54] O. Rippel, S. Nair, C. Lew, S. Branson, A. G. Anderson, and L. Bourdev, "Learned video compression," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 3454–3463.
[55] "Video trace library (VTL) dataset," Accessed: Feb. 1, 2020. [Online]. Available: http://trace.kom.aau.dk/
[56] "Xiph.org video test media dataset," Accessed: Feb. 1, 2020. [Online]. Available: https://media.xiph.org/video/derf//
[57] "The h.264/avc reference software," Accessed: Oct. 30, 2018. [Online]. Available: http://iphome.hhi.de/suehring/
[58] "Hevc test model (HM)," Accessed: Oct. 30, 2018. [Online]. Available: https://hevc.hhi.fraunhofer.de/HM-doc/
[59] "x264, the best H.264/AVC encoder," Accessed: Oct. 30, 2018. [Online]. Available: https://www.videolan.org/developers/x264.html
[60] "x265 HEVC encoder / H.265 video codec," Accessed: Oct. 30, 2018. [Online]. Available: http://x265.org



Guo Lu received the BS degree in electronic engineering from the Ocean University of China, Shandong, China, in 2014. He is currently working toward the PhD degree from the Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University, Shanghai, China. His current research interests include video coding and processing.



Xiaoyun Zhang received the BS and MS degrees in applied mathematics from Xian Jiaotong University, in 1998 and 2001, and the PhD degree in pattern recognition from Shanghai Jiao Tong University, China, in 2004. Her PhD thesis has been nominated as National 100 Best PhD Theses of China. Since 2011, she has been an associate professor at Shanghai Jiao Tong University, and was as a visiting scholar with Harvard Medical School in 2017. Her research interests include computer vision, image and video processing, machine learning, medical image analysis, and computer assisted intervention.

Wanli Ouyang (Senior Member, IEEE) received the PhD degree from the Department of Electronic Engineering, The Chinese University of Hong Kong. Since 2017, he is a senior lecturer with the University of Sydney. His research interests include image processing, computer vision, and pattern recognition.

Li Chen received the BS and MS degrees in electrical engineering from Northwestern Polytechnical University at Xi'an of China, in 1998 and 2000, and the PhD degree in electrical engineering from Shanghai Jiao Tong University, China, in 2006. His current research interests include Image and Video Processing, DSP and VLSI for Image and video processing.

Zhiyong Gao received the BS and MS degrees in electrical engineering from the Changsha Institute of Technology (CIT), Changsha, China, in 1981 and 1984, respectively, and the PhD degree from Tsinghua University, Beijing, China, in 1989. From 1994 to 2010, he took several senior technical positions in England, including a Principal Engineer with Snell & Wilcox, Petersfield, United Kingdom, from 1995 to 2000, a Video Architect with 3DLabs, Egham, United Kingdom, from 2000 to 2001, a Consultant Engineer with Sony European Semiconductor Design Center, Basingstoke, United Kingdom, from 2001 to 2004, and a Digital Video Architect with Imagination Technologies, Kings Langley, United Kingdom, from 2004 to 2010. Since 2010, he has been a professor with Shanghai Jiao Tong University. His research interests include video processing and its implementation, video coding, digital TV, and broadcasting.

Dong Xu (Fellow, IEEE) received the BE and PhD degrees from the University of Science and Technology of China, in 2001 and 2005, respectively. While working toward the PhD degree, he was an intern with Microsoft Research Asia, Beijing, China, and a research assistant with the Chinese University of Hong Kong, Shatin, Hong Kong, for more than two years. He was a post-doctoral research scientist with Columbia University, New York, NY, for one year. He worked as a faculty member with Nanyang Technological University, Singapore. Currently, he is a professor and chair in Computer Engineering with the School of Electrical and Information Engineering, the University of Sydney, Australia. His current research interests include computer vision, statistical learning, and multimedia content analysis. He was the co-author of a paper that won the Best Student Paper award in the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) in 2010, and a paper that won the Prize Paper Award in IEEE Transactions on Multimedia (T-MM), in 2014.