This CVPR 2020 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

M-LVC: Multiple Frames Prediction for Learned Video Compression

Jianping Lin Dong Liu* Houqiang Li Feng Wu
CAS Key Laboratory of Technology in Geo-Spatial Information Processing and Application System,
University of Science and Technology of China, Hefei 230027, China
ljp105@mail.ustc.edu.cn, {dongeliu,lihq,fengwu}@ustc.edu.cn

Abstract

We propose an end-to-end learned video compression scheme for low-latency scenarios. Previous methods are limited in using the previous one *frame as reference. Our method introduces the usage of the previous* multiple *frames as references. In our scheme, the motion vector (MV) field is calculated between the current frame and the previous one. With multiple reference frames and associated multiple MV fields, our designed network can generate more accurate prediction of the current frame, yielding less residual. Multiple reference frames also help generate MV prediction, which reduces the coding cost of MV field. We use two deep auto-encoders to compress the residual and the MV, respectively. To compensate for the compression error of the auto-encoders, we further design a MV refinement network and a residual refinement network, taking use of the multiple reference frames as well. All the modules in our scheme are jointly optimized through a single rate-distortion loss function. We use a step-by-step training strategy to optimize the entire scheme. Experimental results show that the proposed method outperforms the existing learned video compression methods for low-latency mode. Our method also performs better than H.265 in both PSNR and MS-SSIM. Our code and models are publicly available.*

1. Introduction

Video contributes to 75% of all Internet traffic in 2017, and the percent is expected to reach 82% by 2022 [6]. Compressing video into a smaller size is an urgent requirement to reduce the transmission cost. Currently, Internet video is usually compressed into H.264 [29] or H.265 format [21]. New video coding standards like H.266 and AV1 are upcoming. While new standards promise an improvement in compression ratio, such improvement is accompanied with multiplied encoding complexity. Indeed, all the standards in use or in the way coming follow the same framework, that is motion-compensated prediction, block-based transform, and handcrafted entropy coding. The framework has been inherited for over three decades, and the development within the framework is gradually saturated.

Recently, a series of studies try to build brand-new video compression schemes on top of trained deep networks. These studies can be divided into two classes according to their targeted scenarios. As for the first class, Wu *et al*. proposed a recurrent neural network (RNN) based approach for interpolation-based video compression [30], where the motion information is achieved by the traditional block-based motion estimation and is compressed by an image compression method. Later on, Djelouah *et al*. also proposed a method for interpolation-based video compression, where the interpolation model combines motion information compression and image synthesis, and the same auto-encoder is used for image and residual [7]. Interpolation-based compression uses the previous and the subsequent frames as references to compress the current frame, which is valid in random-access scenarios like playback. However, it is less applicable for low-latency scenarios like live transmission.

The second class of studies target low-latency case and restrict the network to use merely temporally previous frames as references. For example, Lu *et al*. proposed DVC, an end-to-end deep video compression model that jointly learns motion estimation, motion compression, motion compensation, and residual compression functions [14]. In this model, only one previous frame is used for motion compensation, which may not fully exploit the temporal correlation in video frames. Rippel *et al*. proposed another video compression model, which maintains a latent state to memorize the information of the previous frames [18]. Due to the presence of the latent state, the model is difficult to train and sensitive to transmission error.

In this paper, we are interested in low-latency scenarios and propose an end-to-end learned video compression scheme. Our key idea is to use the previous *multiple* frames

*This work was supported by the National Key Research and Development Program of China under Grant 2018YFA0701603, and by the Natural Science Foundation of China under Grants 61931014 and 61772483. Code and models are available at https://github.com/JianpingLin/M-LVC_CVPR2020. *(Corresponding author: Dong Liu.)*

as references. Compared to DVC, which uses only one reference frame, our used multiple reference frames enhance the prediction twofold. First, given multiple reference frames and associated multiple motion vector (MV) fields, it is possible to derive multiple hypotheses for predicting the current frame; combination of the hypotheses provides an ensemble. Second, given multiple MV fields, it is possible to extrapolate so as to predict the following MV field; using the MV prediction can reduce the coding cost of MV field. Therefore, our method is termed Multiple frames prediction for Learned Video Compression (M-LVC). Note that in [18], the information of the previous multiple frames is *implicitly* used to predict the current frame through the latent state; but in our scheme, the multiple frames prediction is *explicitly* addressed. Accordingly, our scheme is more scalable (*i.e*. can use more or less references), more interpretable (*i.e*. the prediction is fulfilled by motion compensation), and easier to train per our observation.

Moreover, in our scheme, we design a MV refinement network and a residual refinement network. Since we use a deep auto-encoder to compress MV (resp. residual), the compression is lossy and incurs error in the decoded MV (resp. residual). The MV (resp. residual) refinement network is used to compensate for the compression error and to enhance the reconstruction quality. We also take use of the multiple reference frames and/or associated multiple MV fields in the residual/MV refinement network.

In summary, our technical contributions include:

- We introduce four effective modules into end-to-end learned video compression: multiple frame-based MV prediction, multiple frame-based motion compensation, MV refinement, and residual refinement. Ablation study demonstrates the gain achieved by these modules.

- We use a single rate-distortion loss function, *together with* a step-by-step training strategy, to jointly optimize all the modules in our scheme.

- We conduct extensive experiments on different datasets with various resolutions and diverse content. Our method outperforms the existing learned video compression methods for low-latency mode. Our method performs better than H.265 in both PSNR and MS-SSIM.

2. Related Work

2.1. Learned Image Compression

Recently, deep learning-based image compression methods have achieved great progress [2, 3, 11, 15, 24, 25]. Instead of relying on handcrafted techniques like in conventional image codecs, such as JPEG [26], JPEG2000 [20], and BPG [4], new methods can learn a non-linear transform from data and estimate the probabilities required for entropy coding in an end-to-end manner. In [11, 24, 25], Long Short Term Memory (LSTM) based auto-encoders are used to progressively encode the difference between the original image and the reconstructed image. In addition, there are some studies utilizing convolutional neural network (CNN) based auto-encoders to compress images [2, 3, 15, 23]. For example, Ballé *et al*. [2] introduced a non-linear activation function, generalized divisive normalization (GDN), into CNN-based auto-encoder and estimated the probabilities of latent representations using a fully-connected network. This method outperformed JPEG2000. It does not take into account the input-adaptive entropy model. Ballé *et al*. later in [3] introduced an input-adaptive entropy model by using a zero-mean Gaussian distribution to model each latent representation and the standard deviations are predicted by a parametric transform. More recently, Minnen *et al*. [15] further improved the above input-adaptive entropy model by integrating a context-adaptive model; their method outperformed BPG. In this paper, the modules for compressing the motion vector and the residual are based on the image compression methods in [2, 3]. We remark that new progress on learned image compression models can be easily integrated into our scheme.

2.2. Learned Video Compression

Compared with learned image compression, related work for learned video compression is much less. In 2018, Wu *et al*. proposed a RNN-based approach for interpolation-based video compression [30]. They first use an image compression model to compress the key frames, and then generate the remaining frames using hierarchical interpolation. The motion information is extracted by traditional block-based motion estimation and encoded by a traditional image compression method. Han *et al*. proposed to use variational auto-encoders (VAEs) for compressing sequential data [8]. Their method jointly learns to transform the original video into lower-dimensional representations and to entropy code these representations according to a temporally-conditioned probabilistic model. However, their model is limited to low-resolution video. More recently, Djelouah *et al*. proposed a scheme for interpolation-based video compression, where the motion and blending coefficients are directly decoded from latent representations and the residual is directly computed in the latent space [7]. But the interpolation model and the residual compression model are not jointly optimized.

While the above methods are designed for random-access mode, some other methods have been developed for low-latency mode. For example, Lu *et al*. proposed to replace the modules in the traditional video compression framework with CNN-based components, *i.e*. motion es-

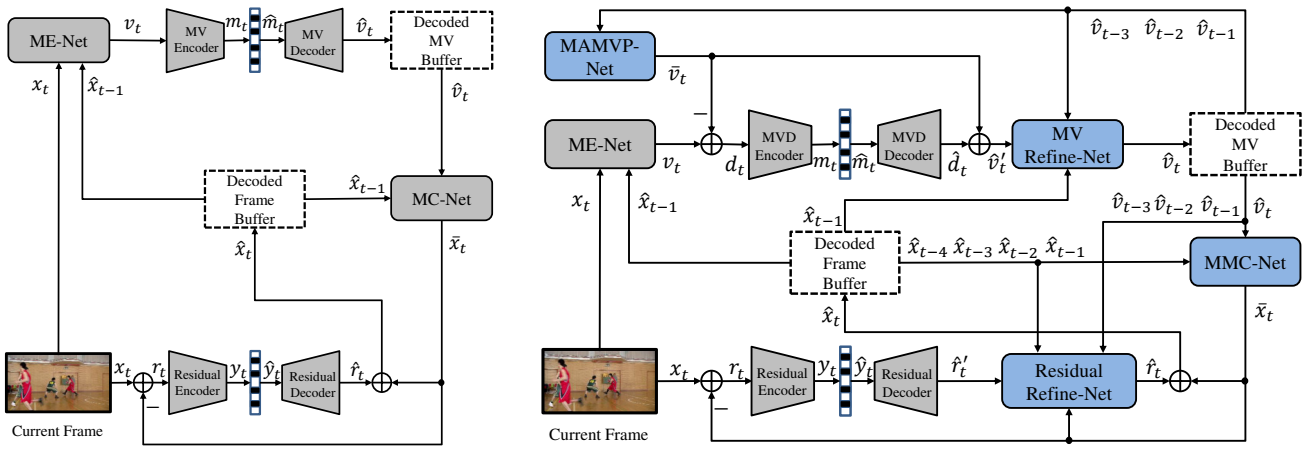


Figure 1. (a) The scheme of DVC [14]. (b) Our scheme. Compared to DVC, our scheme has four new modules that are highlighted in blue. In addition, our Decoded Frame Buffer stores multiple previously decoded frames as references. Our Decoded MV Buffer also stores multiple decoded MV fields. Four reference frames are depicted in the figure, which is the default setting in this paper.

timation, motion compression, motion compensation, and residual compression [14]. Their model directly compresses the motion information, and uses only one previous frame as reference for motion compensation. Rippel *et al*. proposed to utilize the information of multiple reference frames through maintaining a latent state [18]. Due to the presence of the latent state, their model is difficult to train and sensitive to transmission error. Our scheme is also tailored for low-latency mode and we will compare to [14] more specifically in the following.

3. Proposed Method

Notations. Let $\mathcal{V} = \{x_1, x_2, \dots, x_t, \dots\}$ denotes the original video sequence. x_t, $\bar{x}_t$, and $\hat{x}_t$ represent the original, predicted, and decoded/reconstructed frames at time step t, respectively. r_t is the residual between the original frame x_t and the predicted frame $\bar{x}_t$. $\hat{r}'_t$ represents the residual reconstructed by the residual auto-encoder, and $\hat{r}_t$ is the final decoded residual. In order to remove the temporal redundancy between video frames, we use pixel-wise motion vector (MV) field based on optical flow estimation. v_t, $\bar{v}_t$, and $\hat{v}_t$ represent the original, predicted, and decoded MV fields at time step t, respectively. d_t is the MV difference (MVD) between the original MV v_t and the predicted MV $\bar{v}_t$. $\hat{d}_t$ is the MVD reconstructed by the MVD auto-encoder, and $\hat{v}'_t$ represents the reconstructed MV by adding $\hat{d}_t$ to $\bar{v}_t$. Since auto-encoder represents transform, the residual r_t and the MVD d_t are transformed to y_t and m_t. $\hat{y}_t$ and $\hat{m}_t$ are the corresponding quantized versions, respectively.

3.1. Overview of the Proposed Method

Fig. 1 presents the scheme of DVC [14] and our scheme for a side-by-side comparison. Our scheme introduces four new modules, which are all based on multiple reference frames. The specific compression workflow of our scheme is introduced as follows.

Step 1. Motion estimation and prediction. The current frame x_t and the reference frame $\hat{x}_{t-1}$ are fed into a motion estimation network (ME-Net) to extract the motion information v_t. In this paper, the ME-Net is based on the optical flow network FlowNet2.0 [10], which is at the state of the art. Instead of directly encoding the pixel-wise MV field v_t like in Fig. 1 (a), which incurs a high coding cost, we propose to use a MV prediction network (MAMVP-Net) to predict the current MV field, which can largely remove the temporal redundancy of MV fields. More information is provided in Section 3.2.

Step 2. Motion compression and refinement. After motion prediction, we use the MVD encoder-decoder network to encode the difference d_t between the original MV v_t and the predicted MV $\bar{v}_t$. Here the network structure is similar to that in [2]. This MVD encoder-decoder network can further remove the spatial redundancy present in d_t. Specifically, d_t is first non-linearly mapped into the latent representations m_t, and then quantized to $\hat{m}_t$ by a rounding operation. The probability distributions of $\hat{m}_t$ are then estimated by the CNNs proposed in [2]. In the inference stage, $\hat{m}_t$ is entropy coded into a bit stream using the estimated distributions. Then, $\hat{d}_t$ can be reconstructed from the entropy decoded $\hat{m}_t$ by the non-linear inverse transform. Since the decoded $\hat{d}_t$ contains error due to quantization, especially at low bit rates, we propose to use a MV refinement network (MV Refine-Net) to reduce quantization error and improve the quality. After that, the refined MV $\hat{v}_t$ is cached in the decoded MV buffer for next frames coding. More details are presented in Section 3.3.

Step 3. Motion compensation. After reconstructing the MV, we use a motion compensation network (MMC-Net) to obtain the predicted frame $\bar{x}_t$. Instead of only using one reference frame for motion compensation like in Fig. 1 (a), our MMC-Net can generate a more accurate prediction frame by using multiple reference frames. More information is provided in Section 3.4.

Step 4. Residual compression and refinement. After motion compensation, the residual encoder-decoder network is used to encode the residual r_t between the original frame x_t and the predicted frame $\bar{x}_t$. The network structure is similar to that in [3]. This residual encoder-decoder network can further remove the spatial redundancy present in r_t by a powerful non-linear transform, which is also used in DVC [14] because of its effectiveness. Similar to the d_t compression, the residual r_t is first transformed into y_t, and then quantized to $\hat{y}_t$. The probability distributions of $\hat{y}_t$ are then estimated by the CNNs proposed in [3]. In the inference stage, $\hat{y}_t$ is entropy coded into a bit stream using the estimated distributions. Then, $\hat{r}'_t$ can be reconstructed from the entropy decoded $\hat{y}_t$ by the non-linear inverse transform. The decoded $\hat{r}'_t$ contains quantization error, so we propose to use a residual refinement network (Residual Refine-Net) to reduce quantization error and enhance the quality. The details are presented in Section 3.5.

Step 5. Frame reconstruction. After refining the residual, the reconstructed frame $\hat{x}_t$ can be obtained by adding $\hat{r}_t$ to the predicted frame $\bar{x}_t$. $\hat{x}_t$ is then cached in the decoded frame buffer for next frames coding.

3.2. Multi-scale Aligned MV Prediction Network

To address large and complex motion between frames, we propose a Multi-scale Aligned MV Prediction Network (MAMVP-Net), shown in Fig. 2. We use the previous three reconstructed MV fields, *i.e.* $\hat{v}_{t-3}$, $\hat{v}_{t-2}$, and $\hat{v}_{t-1}$, to obtain the MV prediction $\bar{v}_t$. More or less MV fields may be used depending on the size of the Decoded MV Buffer.

As shown in Fig. 2 (a), we first generate a multi-level feature pyramid for each previous reconstructed MV field, using a multi-scale feature extraction network (four levels are used for example),

$$\{f^l_{\hat{v}_{t-i}} | l = 0, 1, 2, 3\} = H_{mf}(\hat{v}_{t-i}), i = 1, 2, 3 \quad (1)$$

where $f^l_{\hat{v}_{t-i}}$ represents the features of $\hat{v}_{t-i}$ at the l-th level. Second, considering the previous reconstructed MV fields contain compression error, we choose to warp the feature pyramids of $\hat{v}_{t-3}$ and $\hat{v}_{t-2}$, instead of the MV fields themselves, towards $\hat{v}_{t-1}$ via:

$$\begin{aligned} f^{l,w}_{\hat{v}_{t-3}} &= Warp(f^l_{\hat{v}_{t-3}}, \hat{v}^l_{t-1} + Warp(\hat{v}^l_{t-2}, \hat{v}^l_{t-1})) \\ f^{l,w}_{\hat{v}_{t-2}} &= Warp(f^l_{\hat{v}_{t-2}}, \hat{v}^l_{t-1}), l = 0, 1, 2, 3 \end{aligned} \quad (2)$$

where $f^{l,w}_{\hat{v}_{t-3}}$ and $f^{l,w}_{\hat{v}_{t-2}}$ are the warped features of $\hat{v}_{t-3}$ and $\hat{v}_{t-2}$ at the l-th level. $\hat{v}^l_{t-1}$ and $\hat{v}^l_{t-2}$ are the down-sampled versions of $\hat{v}_{t-1}$ and $\hat{v}_{t-2}$ at the l-th level. $Warp$ stands for bilinear interpolation-based warping. Note that feature domain warping has been adopted in previous work because of its effectiveness, such as in [16] for video frame interpolation and in [22] for optical flow generation. Third, we use a pyramid network to predict the current MV field from coarse to fine based on the feature pyramid of $\hat{v}_{t-1}$ and the warped feature pyramids of $\hat{v}_{t-2}$ and $\hat{v}_{t-3}$. As shown in Fig. 2 (b), the predicted MV field $\bar{v}^l_t$ and the predicted features $f^l_{\bar{v}_t}$ at the l-th level can be obtained via:

$$\bar{v}^l_t, f^l_{\bar{v}_t} = H_{mvp}(\bar{v}^{l+1,u}_t, f^{l+1,u}_{\bar{v}_t}, f^l_{\hat{v}_{t-1}}, f^{l,w}_{\hat{v}_{t-2}}, f^{l,w}_{\hat{v}_{t-3}}) \quad (3)$$

where $\bar{v}^{l+1,u}_t$ and $f^{l+1,u}_{\bar{v}_t}$ are the $2\times$ up-sampled MV field and features from those at the previous $(l+1)$-th level using bilinear. This process is repeated until the desired 0-th level, resulting in the final predicted MV field $\bar{v}_t$.

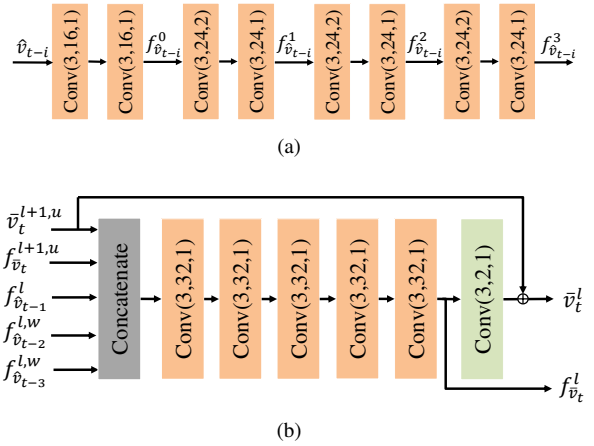


Figure 2. The multi-scale aligned MV prediction network. Conv(3,16,1) denotes the hyper-parameters of a convolutional layer: kernel size is 3×3, output channel number is 16, and stride is 1. Each convolutional layer is equipped with a leaky ReLU except the one indicated by green. (a) Multi-scale feature extraction part. $2\times$ down-sampling is performed by a convolutional layer with a stride of 2, and i is 0, 1, 2. (b) MV prediction part at the l-th level. l is 0, 1, 2, 3, and the network at the 3-th level does not condition on the previous level.

3.3. MV Refinement Network

After MVD compression, we can reconstruct the MV field $\hat{v}'_t$ by adding the decoded MVD $\hat{d}_t$ to the predicted MV $\bar{v}_t$. But $\hat{v}'_t$ contains compression error caused by quantization, especially at low bit rates. For example, we found there are many zeros in $\hat{d}_t$, as zero MVD requires less bits to encode. A similar result is also reported in DVC [14] when compressing the MV field. But such zero MVD incurs inaccurate motion compensation. Therefore, we propose to use a MV refinement network (MV Refine-Net) to reduce compression error and improve the accuracy of reconstructed

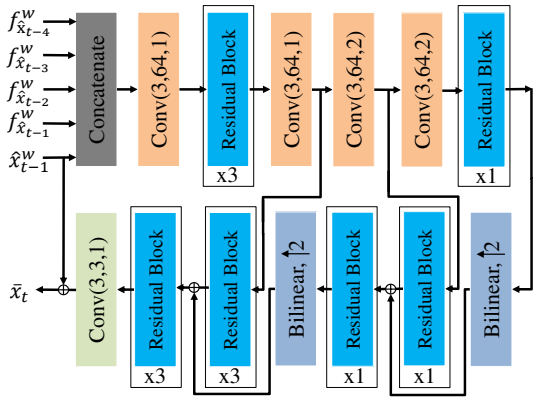


Figure 3. The motion compensation network. Each convolutional layer outside residual blocks is equipped with a leaky ReLU except the last layer (indicated by green). Each residual block consists of two convolutional layers, which are configured as follows: kernel size is 3×3, output channel number is 64, the first layer has ReLU.

MV. As shown in Fig. 1 (b), we use the previous three reconstructed MV fields, *i.e.* $\hat{v}_{t-3}$, $\hat{v}_{t-2}$, and $\hat{v}_{t-1}$, and the reference frame $\hat{x}_{t-1}$ to refine $\hat{v}'_t$. Using the previous multiple reconstructed MV fields can more accurately predict the current MV, and then help on refinement. The reason for using $\hat{x}_{t-1}$ is that the following motion compensation module will depend on the refined $\hat{v}_t$ and $\hat{x}_{t-1}$ to obtain the predicted frame, so $\hat{x}_{t-1}$ can be a guidance to help refine $\hat{v}'_t$. According to our experimental results (Section 4.3), feeding $\hat{x}_{t-1}$ into the MV refinement network does improve the compression efficiency. More details of the MV Refine-Net can been found in the supplementary.

3.4. Motion Compensation Network with Multiple Reference Frames

In traditional video coding schemes, the motion compensation using multiple reference frames is adopted in H.264/AVC [29], and inherited by the following standards. For example, some coding blocks may use a weighted average of two different motion-compensated predictions from different reference frames, which greatly improves the compression efficiency. Besides, in recent work for video super-resolution, multiple frames methods are also observed much better than those based on a single frame [9, 13, 27]. Therefore, we propose to use multiple reference frames for motion compensation in our scheme.

The network architecture is shown in Fig. 3. In this module, we use the previous four reference frames, *i.e.* $\hat{x}_{t-4}$, $\hat{x}_{t-3}$, $\hat{x}_{t-2}$ and $\hat{x}_{t-1}$ to obtain the predicted frame $\bar{x}_t$. More or less reference frames can be used depending on the size of the Decoded Frame Buffer. First, we use a two-layer CNN to extract the features of each reference frame. Then, the extracted features and $\hat{x}_{t-1}$ are warped towards the current frame via:

$$\begin{aligned}
\hat{v}^w_{t-k} &= Warp(\hat{v}_{t-k}, \hat{v}_t + \sum_{l=1}^{k-1} \hat{v}^w_{t-l}), k = 1, 2, 3 \\
\hat{x}^w_{t-1} &= Warp(\hat{x}_{t-1}, \hat{v}_t) \\
f^w_{\hat{x}_{t-i}} &= Warp(f_{\hat{x}_{t-i}}, \hat{v}_t + \sum_{k=1}^{i-1} \hat{v}^w_{t-k}), i = 1, 2, 3, 4
\end{aligned} \tag{4}$$

where $\hat{v}^w_{t-k}$ is the warped version of $\hat{v}_{t-k}$ towards $\hat{v}_t$, and $f^w_{\hat{x}_{t-i}}$ is the warped feature of $\hat{x}_{t-i}$. Finally, as Fig. 3 shows, the warped features and frames are fed into a CNN to obtain the predicted frame,

$$\bar{x}_t = H_{mc}(f^w_{\hat{x}_{t-4}}, f^w_{\hat{x}_{t-3}}, f^w_{\hat{x}_{t-2}}, f^w_{\hat{x}_{t-1}}, \hat{x}^w_{t-1}) + \hat{x}^w_{t-1} \tag{5}$$

where the network is based on the U-Net structure [19] and integrates multiple residual blocks.

3.5. Residual Refinement Network

After residual compression, the reconstructed residual $\hat{r}'_t$ contains compression error, especially at low bit rates. Similar to the case of MV Refine-Net, we propose a residual refinement network (Residual Refine-Net) to reduce compression error and improve quality. As shown in Fig. 1 (b), this module utilizes the previous four reference frames, *i.e.* $\hat{x}_{t-4}$, $\hat{x}_{t-3}$, $\hat{x}_{t-2}$ and $\hat{x}_{t-1}$, and the predicted frame $\bar{x}_t$ to refine $\hat{r}'_t$. More details of this network are provided in the supplementary.

3.6. Training Strategy

Loss Function. Our scheme aims to jointly optimize the number of encoding bits and the distortion between the original frame x_t and the reconstructed frame $\hat{x}_t$. We use the following loss function for training,

$$J = D + \lambda R = d(x_t, \hat{x}_t) + \lambda(R_{mvd} + R_{res}) \tag{6}$$

where $d(x_t, \hat{x}_t)$ is the distortion between x_t and $\hat{x}_t$. We use the mean squared error (MSE) as distortion measure in our experiments. R_{mvd} and R_{res} represent the bit rates used for encoding the MVD d_t and the residual r_t, respectively. During training, we do not perform real encoding but instead estimate the bit rates from the entropy of the corresponding latent representations $\hat{m}_t$ and $\hat{y}_t$. We use the CNNs in [2] and [3] to estimate the probability distributions of $\hat{m}_t$ and $\hat{y}_t$, respectively, and then obtain the corresponding entropy. Since $\hat{m}_t$ and $\hat{y}_t$ are the quantized representations and the quantization operation is not differentiable, we use the method proposed in [2], where the quantization operation is replaced by adding uniform noise during training.

Progressive Training. We had tried to train the entire network from scratch, *i.e.* with all the modules except the ME-Net randomly initialized (ME-Net is readily initialized with FlowNet2.0). The results are not satisfactory, as the resulting bitrates are not balanced: too less rate for MVD

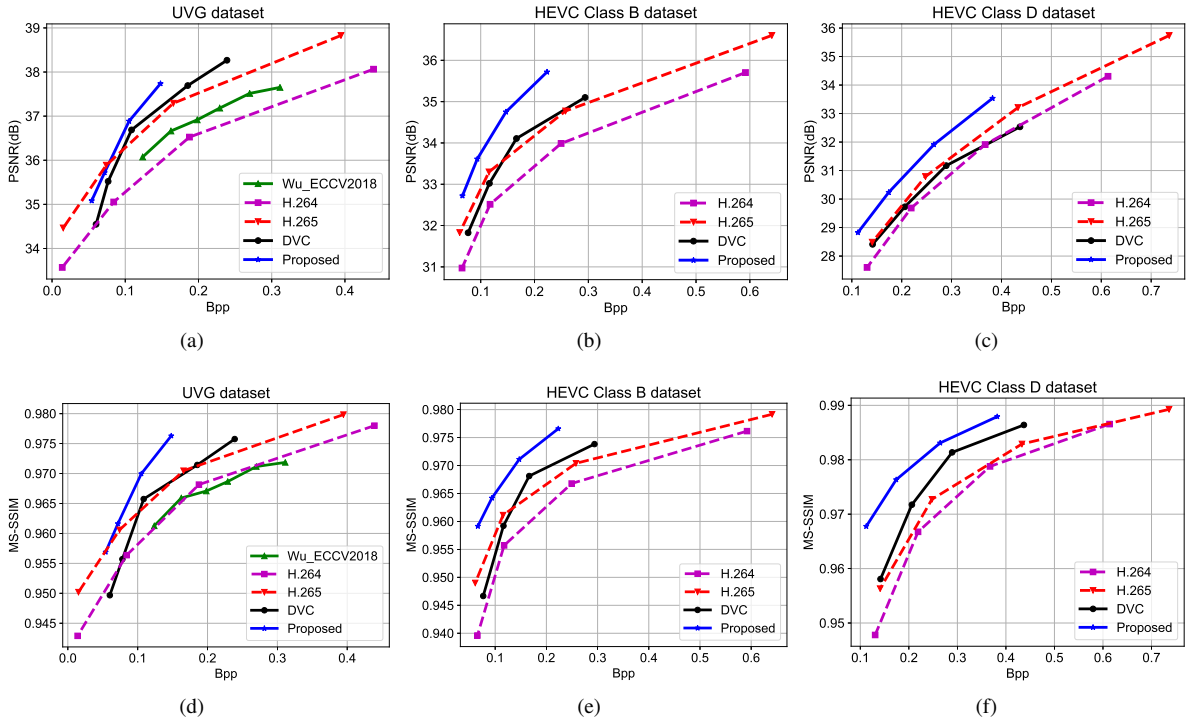


Figure 4. **Overall performance**. The compression results on the three datasets using H.264 [29], H.265 [21], DVC [14], Wu's method [30] and the proposed method. We directly use the results reported in [14] and [30]. The results of H.264 and H.265 are cited from [14]. Wu [30] did not report on HEVC Class B and Class D. Top row: PSNR. Bottom row: MS-SSIM.

and too much rate for residual, and thus the compression results are inefficient (see the experimental results in Section 4.3). To address this problem, we use a step-by-step training strategy. First, we train the network including only the ME-Net and MMC-Net, while the ME-Net is the pre-trained model in [10] and remains unchanged. Then, the MVD and residual encoder-decoder networks are added for training, while the parameters of ME-Net and MMC-Net are fixed. After that, all of the above four modules are jointly fine-tuned. Next, we add the MAMVP-Net, MV Refine-Net and Residual Refine-Net one by one to the training system. Each time when adding a new module, we fix the previously trained modules and learn the new module specifically, and then jointly fine-tune all of them. It is worth noting that many previous studies that use step-by-step training usually adopt a different loss function for each step (*e.g.* [17, 32]), while the loss function remains the same rate-distortion cost in our method.

4. Experiments

4.1. Experimental Setup

Training Data. We use the Vimeo-90k dataset [31], and crop the large and long video sequences into 192×192, 16-frame video clips.

Implementation Details. In our experiments, the coding structure is IPPP... and all the P-frames are compressed by the same network. We do not implement a single image compression network but use H.265 to compress the only I-frame. For the first three P-frames, whose reference frames are less than four, we duplicate the furthest reference frame to achieve the required four frames. We train four models with different λ values $(16, 24, 40, 64)$ for multiple coding rates. The Adam optimizer [12] with the momentum of 0.9 is used. The initial learning rate is $5e{-}5$ for training newly added modules, and $1e{-}5$ in the fine-tuning stages. The learning rate is reduced by a factor of 2 five times during training. Batch size is 8 (*i.e.* 8 cropped clips). The entire scheme is implemented by TensorFlow and trained/tested on a single Titan Xp GPU.

Testing Sequences. The HEVC common test sequences, including 16 videos of different resolutions known as Classes B, C, D, E [5], are used for evaluation. We also use the seven sequences at 1080p from the UVG dataset [1].

Evaluation Metrics. Both PSNR and MS-SSIM [28] are used to measure the quality of the reconstructed frames in comparison to the original frames. Bits per pixel (bpp) is used to measure the number of bits for encoding the representations including MVD and residual.

4.2. Experimental Results

To demonstrate the advantage of our proposed scheme, we compare with existing video codecs, in particular H.264 [29] and H.265 [21]. For easy comparison with DVC, we

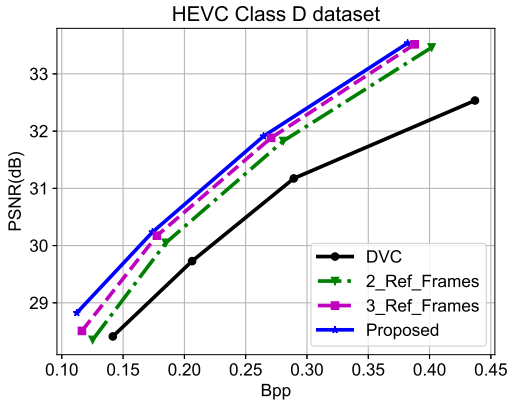


Figure 5. The compression results of using two or three reference frames in our trained models on the HEVC Class D dataset. The proposed model uses four by default and DVC [14] uses only one.

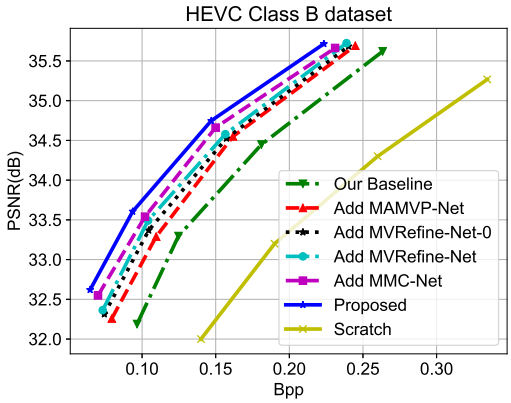


Figure 6. **Ablation study.** The compression results of the following settings on the HEVC Class B dataset. (1) `Our Baseline`: The network contains ME-Net, MC-Net with only one reference frame, and the MV and residual encoder-decoder networks. (2) `Add MAMVP-Net`: The MAMVP-Net is added to (1). (3) `Add MVRefine-Net`: The MV Refine-Net is added to (2). (4) `Add MVRefine-Net-0`: $f_{\hat{x}_{t-1}}$ is removed from the MV Refine-Net in (3). (5) `Add MMC-Net`: The MC-Net with one reference frame in (3) is replaced by the MMC-Net with multiple reference frames. (6) `Proposed`: The Residual Refine-Net is added to (5). (7) `Scratch`: Training (6) from scratch.

directly cite the compression results of H.264 and H.265 reported in [14]. The results of H.264 and H.265 default settings can be found in the supplementary.

In addition, we compare with several state-of-the-art learned video compression methods, including Wu_ECCV2018 [30] and DVC [14]. To the best of our knowledge, DVC [14] reports the best compression performance in PSNR among the learning-based methods for low-latency mode.

Fig. 4 presents the compression results on the UVG dataset and the HEVC Class B and Class D datasets. It can be observed that our method outperforms the learned video compression methods DVC [14] and Wu_ECCV2018 [30] by a large margin. On the HEVC Class B dataset, our method achieves about 1.2dB coding gain than DVC at the same bpp of 0.226. When compared with the traditional codec H.265, our method has achieved better compression performance in both PSNR and MS-SSIM. The gain in MS-SSIM seems more significant. It is worth noting that our model is trained with the MSE loss, but results show that it also works for MS-SSIM. More experimental results, including HEVC Class C and Class E, comparisons to other methods [7, 18], are given in the supplementary.

4.3. Ablation Study

On the Number of Reference Frames. The number of reference frames is an important hyper-parameter in our scheme. Our used default value is four reference frames and their associated MV fields, which is also the default value in the H.265 reference software. To evaluate the effectiveness of using less reference frames, we conduct a comparison experiment by using two or three reference frames in our trained models. Fig. 5 presents the compression results on the HEVC Class D dataset. As observed, the marginal gain of increasing reference frame is less and less.

Multi-scale Aligned MV Prediction Network. To evaluate its effectiveness, we perform a comparison experiment. The anchor is the network containing the ME-Net, the MC-Net with only one reference frame, and the MV and residual encoder-decoder networks. Here, the MC-Net with only one reference frame is almost identical to the MMC-Net shown in Fig. 3, except for removing $f^w_{\hat{x}_{t-4}}$, $f^w_{\hat{x}_{t-3}}$, $f^w_{\hat{x}_{t-2}}$ from the inputs. This anchor is denoted by `Our Baseline` (the green curve in Fig. 6). The tested network is constructed by adding the MAMVP-Net to `Our Baseline`, and is denoted by `Add MAMVP-Net` (the red curve in Fig. 6). It can be observed that the MAMVP-Net improves the compression efficiency significantly, achieving about $0.5 \sim 0.7$ dB gain at the same bpp. In Fig. 7, we visualize the intermediate results when compressing the Kimono sequence using `Add MAMVP-Net` model. Fig. 8 shows the corresponding probability distributions of MV magnitudes for v_6 and d_6. It is observed that the magnitude of MV to be encoded is greatly reduced by using our MAMVP-Net. Quantitatively, it needs 0.042bpp for encoding the original MV v_6 using `Our Baseline` model, while it needs 0.027bpp for encoding the MVD d_6 using `Add MAMVP-Net` model. Therefore, our MAMVP-Net can largely reduce the bits for encoding MV and thus improve the compression efficiency. More ablation study results can be found in the supplementary.

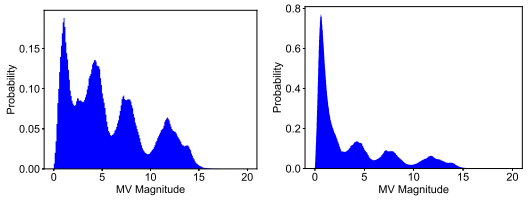
MV Refinement Network. To evaluate the effectiveness, we perform another experiment by adding the MV Refine-Net to `Add MAMVP-Net`, leading to `Add MVRefine-Net` (the cyan curve in Fig. 6). Compared with the compression results of `Add MAMVP-Net`, at the

(a) (b) (c) (d) (e)

Figure 7. Visualized results of compressing the Kimono sequence using `Add MAMVP-Net` model with $\lambda = 16$. (a) The reference frame $\hat{x}_5$. (b) The original frame x_6. (c) The original MV v_6. (d) The predicted MV $\bar{v}_6$. (e) The MVD d_6.

Table 1. Average running time per frame of using our different models for a 320×256 sequence.

Model	`Our Baseline`	`Add MAMVP-Net`	`Add MVRefine-Net`	`Add MMC-Net`	`Proposed`
Encoding Time	0.25s	0.31s	0.34s	0.35s	0.37s
Decoding Time	0.05s	0.11s	0.14s	0.15s	0.17s



(a) (b)

Figure 8. The distribution of MV magnitude. (a) The MV of Fig. 7 (c). (b) The MVD of Fig. 7 (e).

same bpp, the MV Refine-Net achieves a compression gain of about 0.15dB at high bit rates and about 0.4dB at low bit rates. This is understandable as the compression error is more severe when the bit rate is lower. In addition, to evaluate the effectiveness of introducing $\hat{x}_{t-1}$ into the MV Refine-Net, we perform an experiment by removing $f_{\hat{x}_{t-1}}$ from the inputs of the MV Refine-Net (denoted by `Add MVRefine-Net-0`, the black curve in Fig. 6). We can observe that feeding $\hat{x}_{t-1}$ into the MV Refine-Net provides about 0.1dB gain consistently. Visual results of the MV Refine-Net can be found in the supplementary.

Motion Compensation Network with Multiple Reference Frames. To verify the effectiveness, we perform an experiment by replacing the MC-Net (with only one reference frame) in `Add MVRefine-Net` with the proposed MMC-Net using multiple reference frames (denoted by `Add MMC-Net`, the magenta curve in Fig. 6). We can observe that using multiple reference frames in MMC-Net provides about $0.1 \sim 0.25$dB gain. Visual results of the MMC-Net can be found in the supplementary.

Residual Refinement Network. We conduct another experiment to evaluate its effectiveness by adding the Residual Refine-Net to `Add MMC-Net` (denoted by `Proposed`, the blue curve in Fig. 6). We observe that the Residual Refine-Net provides about 0.3dB gain at low bit rates and about 0.2dB gain at high bit rates. Similar to MV Refine-Net, the gain of Residual Refine-Net is higher at lower bit rates because of more compression error. Visual results of the Residual Refine-Net can be found in the supplementary.

Step-by-step Training Strategy. To verify the effectiveness, we perform an experiment by training the `Proposed` model from scratch except the ME-Net initialized by the pre-trained model in [10] (denoted by `Scratch`, the yellow curve in Fig. 6). We can observe that the compression results are very bad. Quantitatively, when compressing the Kimono sequence using `Scratch` model with $\lambda = 16$, the bitrates are very unbalanced: 0.0002bpp for MVD and 0.2431bpp for residual. Our step-by-step training strategy can overcome this.

Encoding and Decoding Time. We use a single Titan Xp GPU to test the inference speed of our different models. The running time is presented in Table 1. We can observe that the MAMVP-Net increases more encoding/decoding time than the other newly added modules. For a 352×256 sequence, the overall encoding (resp. decoding) speed of our `Proposed` model is 2.7fps (resp. 5.9fps). It requires our future work to optimize the network structure for computational efficiency to achieve real-time decoding.

5. Conclusion

In this paper, we have proposed an end-to-end learned video compression scheme for low-latency scenarios. Our scheme can effectively remove temporal redundancy by utilizing multiple reference frames for both motion compensation and motion vector prediction. We also introduce the MV and residual refinement modules to compensate for the compression error and to enhance the reconstruction quality. All the modules in our scheme are jointly optimized by using a single rate-distortion loss function, together with a step-by-step training strategy. Experimental results show that our method outperforms the existing learned video compression methods for low-latency mode. In the future, we anticipate that advanced entropy coding model can further boost the compression efficiency.

References

[1] Ultra video group test sequences. http://ultravideo.cs.tut.fi. accessed: 2018-10-30.

[2] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016.

[3] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018.

[4] F. Bellard. BPG image format (http://bellard.org/bpg/), accessed: 2017-01-30.

[5] Frank Bossen. Common test conditions and software reference configurations. JCTVC-F900, Torino, Italy, July 2011.

[6] VNI Cisco. Cisco visual networking index: Forecast and trends, 2017–2022. *White Paper*, 2018.

[7] Abdelaziz Djelouah, Joaquim Campos, Simone Schaub-Meyer, and Christopher Schroers. Neural inter-frame compression for video coding. In *ICCV*, pages 6421–6429, October 2019.

[8] Jun Han, Salvator Lombardo, Christopher Schroers, and Stephan Mandt. Deep probabilistic video compression. *arXiv preprint arXiv:1810.02845*, 2018.

[9] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. Recurrent back-projection network for video super-resolution. In *CVPR*, pages 3897–3906, June 2019.

[10] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, pages 2462–2470, 2017.

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

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Sheng Li, Fengxiang He, Bo Du, Lefei Zhang, Yonghao Xu, and Dacheng Tao. Fast spatio-temporal residual network for video super-resolution. In *CVPR*, pages 10522–10531, June 2019.

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

[15] David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, pages 10771–10780, 2018.

[16] Simon Niklaus and Feng Liu. Context-aware synthesis for video frame interpolation. In *CVPR*, pages 1701–1710, 2018.

[17] Fitsum A. Reda, Guilin Liu, Kevin J. Shih, Robert Kirby, Jon Barker, David Tarjan, Andrew Tao, and Bryan Catanzaro. SDC-net: Video prediction using spatially-displaced convolution. In *ECCV*, pages 718–733, 2018.

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

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015.

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

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

[22] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR*, pages 8934–8943, 2018.

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

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

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

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

[27] Xintao Wang, Kelvin Chan, Ke Yu, Chao Dong, and Chen Change Loy. EDVR: Video restoration with enhanced deformable convolutional networks. In *CVPR Workshops*, 2019.

[28] Zhou Wang, Eero P. Simoncelli, and Alan C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. IEEE, 2003.

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

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

[31] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T. Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127(8):1106–1125, 2019.

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