Open access • Journal Article • DOI:10.1109/JSTSP.2020.3043590

Learning for Video Compression With Recurrent Auto-Encoder and Recurrent Probability Model — Source link

Ren Yang, Fabian Mentzer, Luc Van Gool, Radu Timofte

Institutions: ETH Zurich

Published on: 01 Feb 2021 - IEEE Journal of Selected Topics in Signal Processing (IEEE)

Topics: Motion compensation, Reference frame, Data compression, Encoder and Probability mass function

Learning for Video Compression with Recurrent Auto-Encoder and Recurrent Probability Model

Ren Yang, *Student Member, IEEE,* Fabian Menzter, *Student Member, IEEE,*
Luc Van Gool, *Member, IEEE,* and Radu Timofte, *Member, IEEE*

***Abstract*—The past few years have witnessed increasing interests in applying deep learning to video compression. However, the existing approaches compress a video frame with only a few number of reference frames, which limits their ability to fully exploit the temporal correlation among video frames. To overcome this shortcoming, this paper proposes a Recurrent Learned Video Compression (RLVC) approach with the Recurrent Auto-Encoder (RAE) and Recurrent Probability Model (RPM). Specifically, the RAE employs recurrent cells in both the encoder and decoder. As such, the temporal information in a large range of frames can be used for generating latent representations and reconstructing compressed outputs. Furthermore, the proposed RPM network recurrently estimates the Probability Mass Function (PMF) of the latent representation, conditioned on the distribution of previous latent representations. Due to the correlation among consecutive frames, the conditional cross entropy can be lower than the independent cross entropy, thus reducing the bit-rate. The experiments show that our approach achieves the state-of-the-art learned video compression performance in terms of both PSNR and MS-SSIM. Moreover, our approach outperforms the default Low-Delay P (LDP) setting of x265 on PSNR, and also has better performance on MS-SSIM than the SSIM-tuned x265 and the slowest setting of x265.**

The code and pre-trained models will be released on the project page: https://github.com/RenYang-home/RLVC.git.

***Index Terms*—Deep learning, video compression, representation learning.**

I. Introduction

NOWADAYS, video contributes to the majority of mobile data traffic [1]. The demands of high resolution and high quality video are also increasing. Therefore, video compression is essential to enable the efficient transmission of video data over the band-limited Internet. Especially, during the COVID-19 pandemic, the increasing data traffic used for video conferencing, gaming and online learning forced Netflix and YouTube to limit video quality in Europe. This further shows the essential impact of improving video compression on today's social development.

During the past decades, several video compression algorithms, such as MPEG [2], H.264 [3] and H.265 [4] were standardized. These standards are handcrafted, and the modules in compression frameworks cannot be jointly optimized. Recently, inspired by the success of Deep Neural Networks (DNN) in advancing the rate-distortion performance of image compression [5]–[7], many deep learning-based video compression approaches [8]–[12] were proposed. In these learned video compression approaches, the whole frameworks are optimized in an end-to-end manner.

The authors are with the Department of Information Technology and Electrical Engineering, ETH Zürich, 8092 Zurich, Switzerland (e-mail: ren.yang@vision.ee.ethz.ch). This work is partly supported by ETH Zurich General Fund (OK) and Amazon AWS Grant.

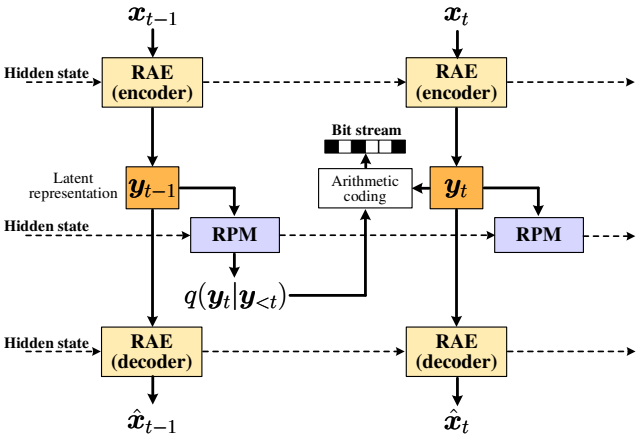


Fig. 1. The recurrent structure in our RLVC approach. In this figure, two time steps are shown as an example.

However, both the existing handcrafted [2]–[4] and learned video compression [8]–[12] approaches utilize non-recurrent structures to compress the sequential video data. As such, only a limited number of references can be used to compress new frames, thus limiting their ability for exploring temporal correlation and reducing redundancy. Adopting a recurrent compression framework enables to fully take advantage of the correlated information in consecutive frames, and thus facilitates video compression. Moreover, in the entropy coding of previous learned approaches [8]–[12], the Probability Mass Functions (PMF) of latent representations are also independently estimated on each frame, ignoring the correlation between the latent representations among neighboring frames. Similar to the reference frames in the pixel domain, fully making use of the correlation in the latent domain benefits the compression of latent representations. Intuitively, the temporal correlation in the latent domain also can be explored in a recurrent manner.

Therefore, this paper proposes a Recurrent Learned Video Compression (RLVC) approach, with the Recurrent Auto-Encoder (RAE) and Recurrent Probability Model (RPM). As shown in Fig. 1, the proposed RLVC approach uses recurrent networks for representing inputs, reconstructing compressed outputs and modeling PMFs for entropy coding. Specifically, the proposed RAE network contains recurrent cells in both the encoder and decoder. Given a sequence of inputs $\{\boldsymbol{x}_t\}_{t=1}^{T}$, the encoder of RAE recurrently generates the latent representations $\{\boldsymbol{y}_t\}_{t=1}^{T}$, and the decoder also reconstructs the

compressed outputs $\{\hat{\boldsymbol{x}}_t\}_{t=1}^T$ from $\{\boldsymbol{y}_t\}_{t=1}^T$ in a recurrent manner. As such, all previous frames can be seen as references for compressing the current one, and therefore our RLVC approach is able to make use of the information in a large number of frames, instead of the very limited reference frames in the non-recurrent approaches [8]–[10], [12].

Furthermore, the proposed RPM network recurrently models the PMF of $\boldsymbol{y}_t$ conditioned on all previous latent representations $\boldsymbol{y}_{<t} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}\}$. Because of the recurrent cell, our RPM network estimates the temporally *conditional* PMF $q(\boldsymbol{y}_t \,|\, \boldsymbol{y}_{<t})$, instead of the *independent* PMF $q(\boldsymbol{y}_t)$ as in previous works [8]–[12]. Due to the temporal correlation among $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t\}$, the (cross) entropy of $\boldsymbol{y}_t$ *conditioned* on the previous information $\boldsymbol{y}_{<t}$ is expected to be lower than the *independent* (cross) entropy. Therefore, our RPM network is able to achieve lower bit-rate to compress $\boldsymbol{y}_t$. As Fig. 1 illustrates, the proposed RAE and RPM networks build up a recurrent video compression framework. The hidden states for representation learning and probability modeling are recurrently transmitted from frame to frame, and therefore the information in consecutive frames can be fully exploited in both the pixel and latent domains for compressing the upcoming frames. This results in efficient video compression.

The contribution of this paper can be summarized as:

- We propose employing the recurrent structure in learned video compression to fully exploit the temporal correlation among a large range of video frames.
- We propose the recurrent auto-encoder to expand the range of reference frames, and propose the recurrent probability model to recurrently estimate the temporally conditional PMF of the latent representations. This way, we achieve the expected bit-rate as the conditional cross entropy, which can be lower than the independent cross entropy in previous non-recurrent approaches.
- The experiments validate the superior performance of the proposed approach to the existing learned video compression approaches, and the ablation studies verify the effectiveness of each recurrent component in our framework.

In the following, Section II presents the related works. The proposed RAE and RPM are introduced in Section III. Then, the experiments in Section IV validate the superior performance of the proposed RLVC approach to the existing learned video compression approaches. Finally, the ablation studies further demonstrate the effectiveness of the proposed RAE and RPM networks, respectively.

II. Related works

Auto-encoders and RNNs. Auto-encoders [13] have been popularly used for representation learning in the past decades. In the field of image processing, there are plenty of auto-encoders proposed for image denoising [14], [15], enhancement [16], [17] and super resolution [18], [19]. Besides, inspired by the development of Recurrent Neural Networks (RNNs) and their applications on sequential data [20], *e.g.*, language modeling [21], [22] and video analysis [23], some recurrent auto-encoders were proposed for representation learning on time-series tasks, such as machine translation [24], [25] and captioning [26], *etc.* Moreover, Srivastava *et al.* [27] proposed learning for video representations using an auto-encoder based on Long Short-Term Memory (LSTM) [28], and verified the effectiveness on classification and action recognition tasks on video. However, as far as we know, there is no recurrent auto-encoder utilized in learned video compression.

Learned image compression. In recent years, there are increasing interests in applying deep auto-encoders in the end-to-end DNN models for learned image compression [5]–[7], [29]–[37]. For instance, Theis *et al.* [32] proposed a compressive auto-encoder for lossy image compression, and reached competitive performance with JPEG 2000 [38]. Later, various probability models were proposed. For instance, Ballé *et al.* [33], [34] proposed the factorized prior [33] and hyper-prior [34] probability models to estimate entropy in the end-to-end DNN image compression frameworks. Later, based on them, Minnen *et al.* [5] proposed the hierarchical prior entropy model to improve the compression efficiency. Besides, Mentzer *et al.* [35] utilized 3D-CNN as the context model for entropy coding, and proposed learning an importance mask to reduce the redundancy in latent representation. Recently, the context-adaptive [6] and the coarse-to-fine hyper-prior [7] entropy models were designed to further advance the rate-distortion performance, and successfully outperform the traditional image codec BPG [39].

Learned video compression. Deep learning is also attracting more and more attention in video compression. To improve the coding efficiency of handcrafted standard, many approaches [40]–[45] were proposed to replace the components in H.265 by DNN. Among them, Liu *et al.* [41] utilized DNN in the fractional interpolation of motion compensation, and Choi *et al.* [42] proposed a DNN model for frame prediction. Besides, [43]–[45] employed DNNs to improve the in-loop filter of H.265. However, these approaches only advance the performance of one particular module, and the video compression frameworks cannot be jointly optimized.

Inspired by the success of learned image compression, some learning-based video compression approaches were proposed [46], [47]. However, [46], [47] still adopt some handcrafted strategies, such as block matching for motion estimation and compensation, and therefore they fail to optimize the whole compression framework in an end-to-end manner. Recently, several end-to-end DNN frameworks have been proposed for video compression [8]–[12], [48], [49]. Specifically, Wu *et al.* [8] proposed predicting frames by interpolation from reference frames, and compressing residual by the image compression model [30]. Later, Lu *et al.* [9] proposed the Deep Video Compression (DVC) approach, which uses optical flow for motion estimation, and utilizes two auto-encoders to compress the motion and residual, respectively. Then, Djelouah *et al.* [11] employs bi-directional prediction in to learned video compression. Liu *et al.* [49] proposed a deep video compression framework with the one-stage flow for motion compensation. Most recently, Yang *et al.* [12] proposed learning for video compression with hierarchical quality layers and adopted a recurrent enhancement network in the deep decoder. Agustsson *et al.* [50] proposed the scale-space flow

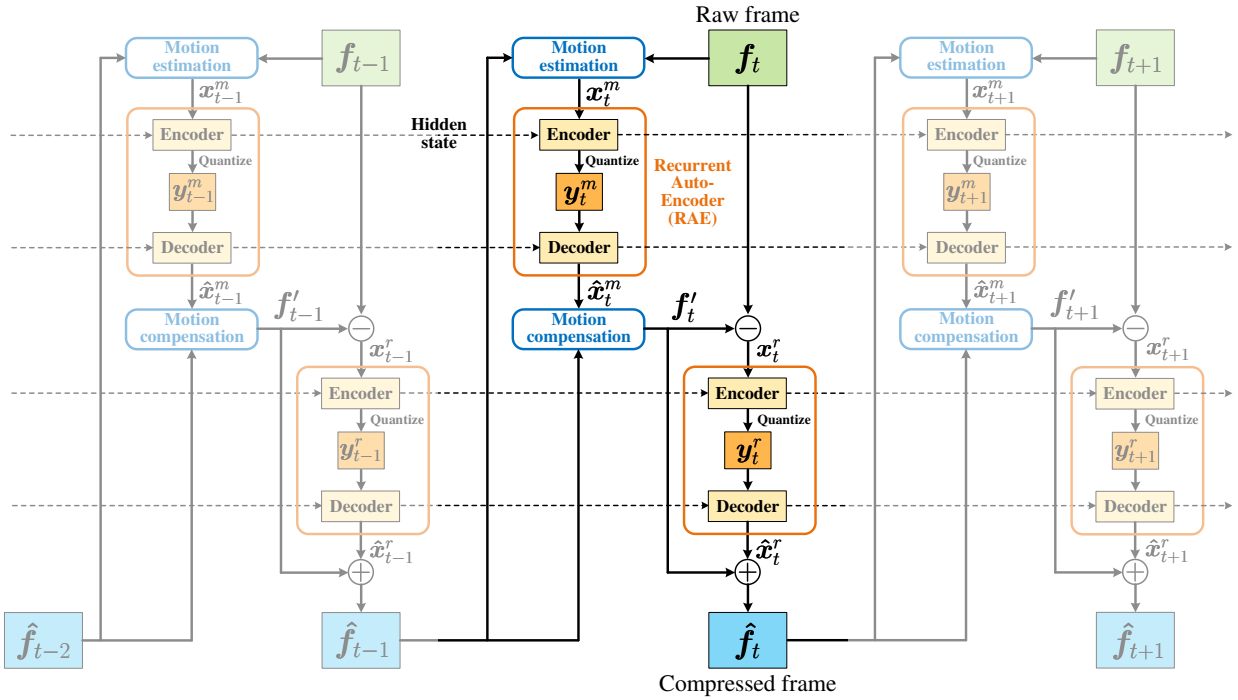


Fig. 2. The framework of our RLVC approach. The details of the proposed RAE are shown in Fig. 3. The proposed RPM, which is illustrated in Fig. 5, is applied on the latent representations $\boldsymbol{y}_t^m$ and $\boldsymbol{y}_t^r$ to estimate their conditional PMF for arithmetic coding.

for learned video compression, which learns to adaptively blur frame if the bilinearly warped frame is not a good prediction. Nevertheless, none of them learns to compress video with a recurrent model. Instead, there are at most two reference frames used in these approaches [8]–[12], [49], and therefore they fail to exploit the temporal correlation in a large number of frames.

Although Habibian *et al.* [48] proposed taking a group of frames as inputs to the 3D auto-encoder, the temporal length is limited as all frames in one group have to fit into GPU memory *at the same time*. Instead, the proposed RLVC network takes as inputs only one frame and the hidden states from the previous frame, and recurrently moves forward. Therefore, we are able to explore larger range of temporal correlation with finite memory. Also, [48] uses a PixelCNN-like network [51] as an auto-regressive probability model, which makes decoding slow. On the contrary, the proposed RPM network benefits our approach to achieve not only more efficient compression but also faster decoding.

III. The Proposed RLVC Approach

A. Framework

The framework of the proposed RLVC approach is shown in Fig. 2. Inspired by traditional video codecs, we utilize motion compensation to reduce the redundancy among video frames, whose effectiveness in learned compression has been proved in previous works [9], [12]. To be specific, we apply the pyramid optical flow network [52] to estimate the temporal motion between the current frame and the previously compressed frame, *e.g.*, $\boldsymbol{f}_t$ and $\hat{\boldsymbol{f}}_{t-1}$. The large receptive field of the pyramid network [52] benefits to handle large and fast motions. Here, we define the raw and compressed frames as $\{\boldsymbol{f}_t\}_{t=1}^T$ and $\{\hat{\boldsymbol{f}}_t\}_{t=1}^T$, respectively. Then, the estimated motion $\boldsymbol{x}_t^m$ is compressed by the proposed RAE, and the compressed motion $\hat{\boldsymbol{x}}_t^m$ is applied for motion compensation. In our framework, we use the same motion compensation method as [9], [12]. In the following, the residual ($\boldsymbol{x}_t^r$) between $\boldsymbol{f}_t$ and the motion compensated frame $\boldsymbol{f}_t'$ can be obtained and compressed by another RAE. Given the compressed residual as $\hat{\boldsymbol{x}}_t^r$, the compressed frame $\hat{\boldsymbol{f}}_t = \boldsymbol{f}_t' + \hat{\boldsymbol{x}}_t^r$ can be reconstructed. The details of the proposed RAE is described in Section III-B.

In our framework, the two RAEs in each frame generate the latent representations of $\boldsymbol{y}_t^m$ and $\boldsymbol{y}_t^r$ for motion and residual compression, respectively. To compress $\boldsymbol{y}_t^m$ and $\boldsymbol{y}_t^r$ into a bit stream, we propose the RPM network to recurrently predict the temporally *conditional* PMFs of $\{\boldsymbol{y}_t^m\}_{t=1}^T$ and $\{\boldsymbol{y}_t^r\}_{t=1}^T$. Due to the temporal relationship among video frames, the *conditional* cross entropy is expected to be lower than the *independent* cross entropy used in non-recurrent approaches [8]–[10], [12]. Hence, utilizing the conditional PMF estimated by our RPM network effectively reduces bit-rate in arithmetic coding [53]. The proposed RPM is detailed in Section III-C.

B. Recurrent Auto-Encoder (RAE)

As mentioned above, we apply two RAEs to compress $\boldsymbol{x}_t^m$ and $\boldsymbol{x}_t^r$. Since the two RAEs share the same architecture, we denote both $\boldsymbol{x}_t^m$ and $\boldsymbol{x}_t^r$ by $\boldsymbol{x}_t$ in this section for simplicity. Recall that in the non-recurrent learned video compression works [9], [10], [12], when compressing the t-th frame, the auto-encoders map the input $\boldsymbol{x}_t$ to a latent representation

$$\tilde{\boldsymbol{y}}_t = E(\boldsymbol{x}_t; \boldsymbol{\theta}_E) \tag{1}$$

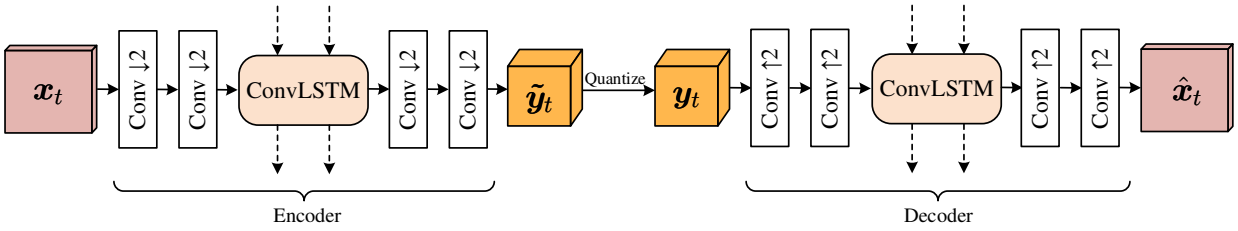


Fig. 3. The architecture of the proposed RAE network. In convolutions layers, ↑ 2 and ↓ 2 indicate up- and down-sampling with the stride of 2, respectively. In RAE, the filter sizes of all convolutional layers are set as 3×3 when compressing motion, and set as 5×5 for residual compression. The filter number of each layer is set as 128.

through an encoder E parametrized with $\boldsymbol{\theta}_E$. Then, the continuous-valued $\tilde{\boldsymbol{y}}_t$ is quantized to the discrete-valued $\boldsymbol{y}_t = \lfloor \tilde{\boldsymbol{y}}_t \rceil$. The compressed output is reconstructed by the decoder from the quantized latent representation, *i.e.*,

$$\hat{\boldsymbol{x}}_t = D(\boldsymbol{y}_t; \boldsymbol{\theta}_D). \tag{2}$$

Taking the inputs of only the current $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$ to the encoder and decoder, they fail to take advantage of the temporal correlation in consecutive frames.

On the contrary, the proposed RAE includes recurrent cells in both the encoder and decoder. The architecture of the RAE network is illustrated in Fig. 3. We follow [34] to use four $2\times$ down-sampling convolutional layers with the activation function of GDN [33] in the encoder of RAE. In the middle of the four convolutional layers, we insert a ConvLSTM [54] cell to achieve the recurrent structure. As such, the information from previous frames flows into the encoder network of the current frame through the hidden states of the ConvLSTM. Therefore, the proposed RAE generates latent representation based on the current *as well as* previous inputs. Similarly, the recurrent decoder in RAE also has a ConvLSTM cell in middle of the four $2\times$ up-sampling convolutional layers with IGDN [33], and thus also reconstructs $\hat{\boldsymbol{x}}_t$ from both the current and previous latent representations. In summary, our RAE network can be formulated as

$$\begin{aligned} \boldsymbol{y}_t &= \lfloor E(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t; \boldsymbol{\theta}_E) \rceil, \\ \hat{\boldsymbol{x}}_t &= D(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t; \boldsymbol{\theta}_D). \end{aligned} \tag{3}$$

In (3), all previous frames can be seen as reference frames for compressing the current frame, and therefore our RLVC approach is able to make use of the information in a large range of frames, instead of the very limited number of reference frames in the non-recurrent approaches [8]–[10], [12].

C. Recurrent Probability Model (RPM)

To compress the sequence of latent representations $\{\boldsymbol{y}_t\}_{t=1}^T$, the RPM network is proposed for entropy coding. First, we use $p(\boldsymbol{y}_t)$ and $q(\boldsymbol{y}_t)$ to denote the true and estimated *independent* PMFs of $\boldsymbol{y}_t$. The expected bit-rate of $\boldsymbol{y}_t$ is then given as the cross entropy

$$H(p, q) = \mathbb{E}_{\boldsymbol{y}_t \sim p}[-\log_2 q(\boldsymbol{y}_t)]. \tag{4}$$

Note that arithmetic coding [53] is able to encode $\boldsymbol{y}_t$ at the bit-rate of the cross entropy with negligible overhead. It can be seen from (4) that if $\boldsymbol{y}_t$ has higher certainty, the bit-rate can be smaller. Due to the temporal relationship among video frames, the distribution of $\boldsymbol{y}_t$ in consecutive frames are correlated. Therefore, conditioned on the information of previous latent representations $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}$, the current $\boldsymbol{y}_t$ is expected to be more certain. That is, defining $p_t(\boldsymbol{y}_t \,|\, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})$ and $q_t(\boldsymbol{y}_t \,|\, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})$ as the true and estimated temporally *conditional* PMF of $\boldsymbol{y}_t$, the *conditional* cross entropy

$$H(p_t, q_t) = \mathbb{E}_{\boldsymbol{y}_t \sim p_t}[-\log_2 q_t(\boldsymbol{y}_t \,|\, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})] \tag{5}$$

can be smaller than the *independent* cross entropy in (4). To achieve the expected bit-rate of (5), we propose the RPM network to recurrently model the *conditional* PMF $q_t(\boldsymbol{y}_t \,|\, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})$.

Specifically, adaptive arithmetic coding [53] allows to change the PMF for each element in $\boldsymbol{y}_t$, and thus we estimate different conditional PMFs $q_{it}(y_{it} \,|\, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})$ for different elements y_{it}. Here, y_{it} is defined as the element at the i-th 3D *location* in $\boldsymbol{y}_t$, and the conditional PMF of $\boldsymbol{y}_t$ can be expressed as

$$q_t(\boldsymbol{y}_t \,|\, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}) = \prod_{i=1}^{N} q_{it}(y_{it} \,|\, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}), \tag{6}$$

in which N denotes the number of 3D positions in $\boldsymbol{y}_t$. As shown in Fig. 4, we model $q_{it}(y_{it} \,|\, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})$ of each element as discretized logistic distribution in our approach. Since the quantization operation in RAE quantizes all $\tilde{y}_{it} \in [y_{it} - 0.5, y_{it} + 0.5)$ to a discrete value y_{it}, the conditional PMF of the quantized y_{it} can be obtained by integrating the continuous logistic distribution [55] from $(y_{it} - 0.5)$ to $(y_{it} + 0.5)$:

$$q_{it}(y_{it} \,|\, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}) = \int_{y_{it}-0.5}^{y_{it}+0.5} \text{Logistic}(y; \mu_{it}, s_{it}) dy, \tag{7}$$

in which the logistic distribution is defined as

$$\text{Logistic}(y; \mu, s) = \frac{\exp(-(y-\mu)/s)}{s(1+\exp(-(y-\mu)/s))^2}, \tag{8}$$

and its integral is the sigmoid distribution, *i.e.*,

$$\int \text{Logistic}(y; \mu, s) dy = \text{Sigmod}(y; \mu, s) + C. \tag{9}$$

Given (7), (8) and (9), the estimated conditional PMF can be simplified as

$$\begin{aligned} q_{it}(y_{it} \,|\, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}) = \text{Sigmoid}(y_{it} + 0.5; \mu_{it}, s_{it}) \\ - \text{Sigmoid}(y_{it} - 0.5; \mu_{it}, s_{it}). \end{aligned} \tag{10}$$

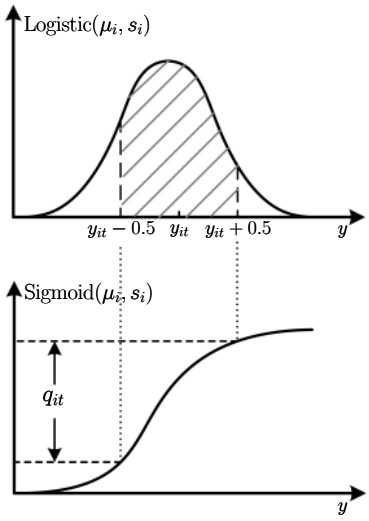


Fig. 4. Modeling the conditional PMF with a discretized logistic distribution.

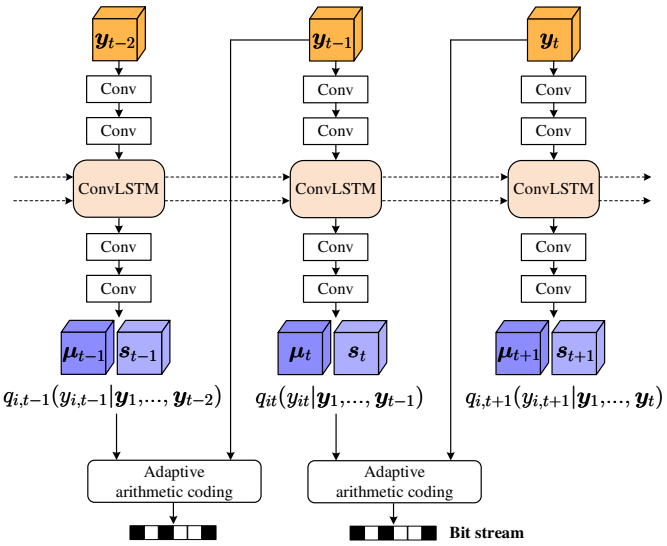


Fig. 5. The architecture of the RPM network, in which all layers have 128 convolutional filters with the size of 3×3.

It can be seen from (10), the conditional PMF at each *location* is modelled with parameters μ_{it} and s_{it}, which are varying for different locations in $\boldsymbol{y}_t$. The RPM network is proposed to recurrently estimate $\boldsymbol{\mu}_t = \{\mu_{it}\}_{i=1}^N$ and $\boldsymbol{s}_t = \{s_{it}\}_{i=1}^N$ in (10). Fig. 5 demonstrates the detailed architecture of our RPM network, which contains a recurrent network P with convolution layers and a ConvLSTM cell in the middle. Due to the recurrent structure, $\boldsymbol{\mu}_t$ and $\boldsymbol{s}_t$ are generated based on all previous latent representations, *i.e.*,

$$\boldsymbol{\mu}_t, \boldsymbol{s}_t = P(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}; \boldsymbol{\theta}_P), \tag{11}$$

where $\boldsymbol{\theta}_P$ represents the trainable parameters in RPM. Because P takes previous latent representations $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}$ as inputs, $\boldsymbol{\mu}_t$ and $\boldsymbol{s}_t$ learn to model the probability of each y_{it} conditioned on $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}$ according to (10). Finally, the conditional PMFs $q_{it}(y_{it} \,|\, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})$ are applied to the adaptive arithmetic coding [53] to encode $\boldsymbol{y}_t$ into a bit stream.

D. Training

In this paper, we utilize the Multi-Scale Structural SIMilarity (MS-SSIM) index and the Peak Signal-to-Noise Ratio (PSNR) to evaluate compression quality, and train two models optimized for MS-SSIM and PSNR, respectively. The distortion D is defined as $1 -$ MS-SSIM when optimizing for MS-SSIM, and as the Mean Square Error (MSE) when training the PSNR model. As Fig. 2 shows, our approach uses the uni-directional Low-Delay P (LDP) structure. We follow [12] to compress the I-frame $\boldsymbol{f}_0$ with the learned image compression method [6] for the MS-SSIM model, and with BPG [39] for the PSNR model. Because of lacking previous latent representation for the first P-frame $\boldsymbol{f}_1$, $\boldsymbol{y}_1^m$ and $\boldsymbol{y}_1^r$ are compressed by the spatial entropy model of [33], with the bit-rate defined as $R_1(\boldsymbol{y}_1^m)$ and $R_1(\boldsymbol{y}_1^r)$, respectively. The following P-frames are compressed with the proposed RPM network. For $t \geq 2$, the actual bit-rate can be calculated as

$$\begin{aligned} R_{\text{RPM}}(\boldsymbol{y}_t) &= -\log_2(q_t(\boldsymbol{y}_t \,|\, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})) \\ &= \sum_{i=1}^{N} -\log_2(q_{it}(y_{it} \,|\, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})), \end{aligned} \tag{12}$$

in which $q_t(\boldsymbol{y}_t \,|\, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})$ is modelled by the proposed RPM according to (6) to (11). Note that, assuming that the distribution of the training set is identical with the true distribution, the actual bit-rate $R_{\text{RPM}}(\boldsymbol{y}_t)$ is expected to be the conditional cross entropy in (5). In our approach, two RPM networks are applied to the latent representations of motion and residual, and their bit-rates are defined as $R_{\text{RPM}}(\boldsymbol{y}_t^m)$ and $R_{\text{RPM}}(\boldsymbol{y}_t^r)$, respectively.

Our RLVC approach is trained on the Vimeo-90k [56] dataset, in which each training sample has 7 frames. The first frame is compressed as the I-frame and the other 6 frames are P-frames. First, we warm up the network on the first P-frame $\boldsymbol{f}_1$ in a progressive manner. At the beginning, the motion estimation network is trained with the loss function of

$$\mathcal{L}_{\text{ME}} = D(\boldsymbol{f}_1, W(\boldsymbol{f}_0, \boldsymbol{x}_1^m)), \tag{13}$$

in which $\boldsymbol{x}_1^m$ is the output of the motion estimation network (as shown in Fig. 2) and W is the warping operation. When L_{ME} is converged, we further include the RAE network for compressing motion and the motion compensation network into training, using the following loss function

$$\mathcal{L}_{\text{MC}} = \lambda \cdot D(\boldsymbol{f}_1, \boldsymbol{f}_1') + R_1(\boldsymbol{y}_1^m). \tag{14}$$

After the convergence of L_{MC}, the whole network is jointly trained on $\boldsymbol{f}_1$ by the loss of

$$\mathcal{L}_1 = \lambda \cdot D(\boldsymbol{f}_1, \hat{\boldsymbol{f}}_1) + R_1(\boldsymbol{y}_1^m) + R_1(\boldsymbol{y}_1^r). \tag{15}$$

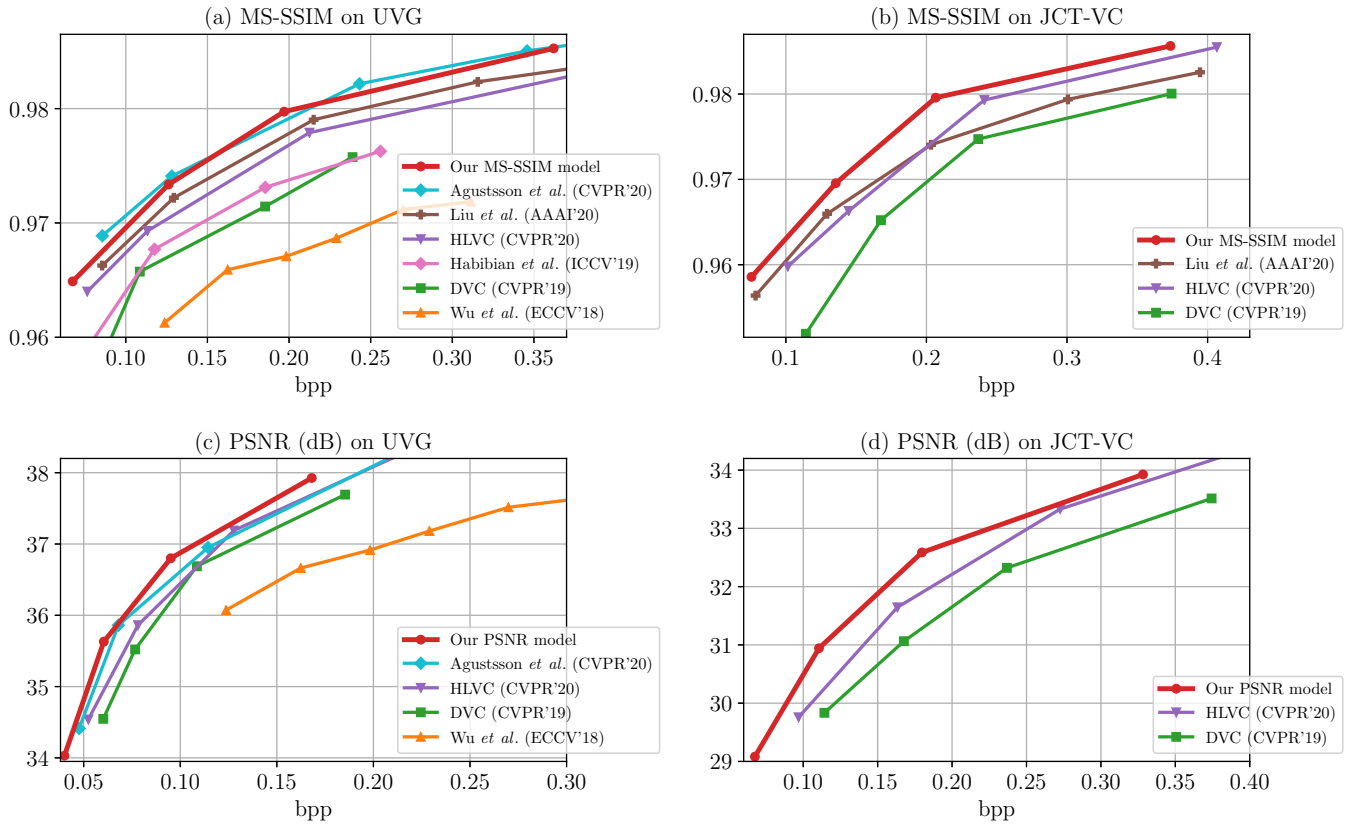


Fig. 6. The rate-distortion performance of our RLVC approach compared with the learned video compression approaches on the UVG and JCT-VC datasets.

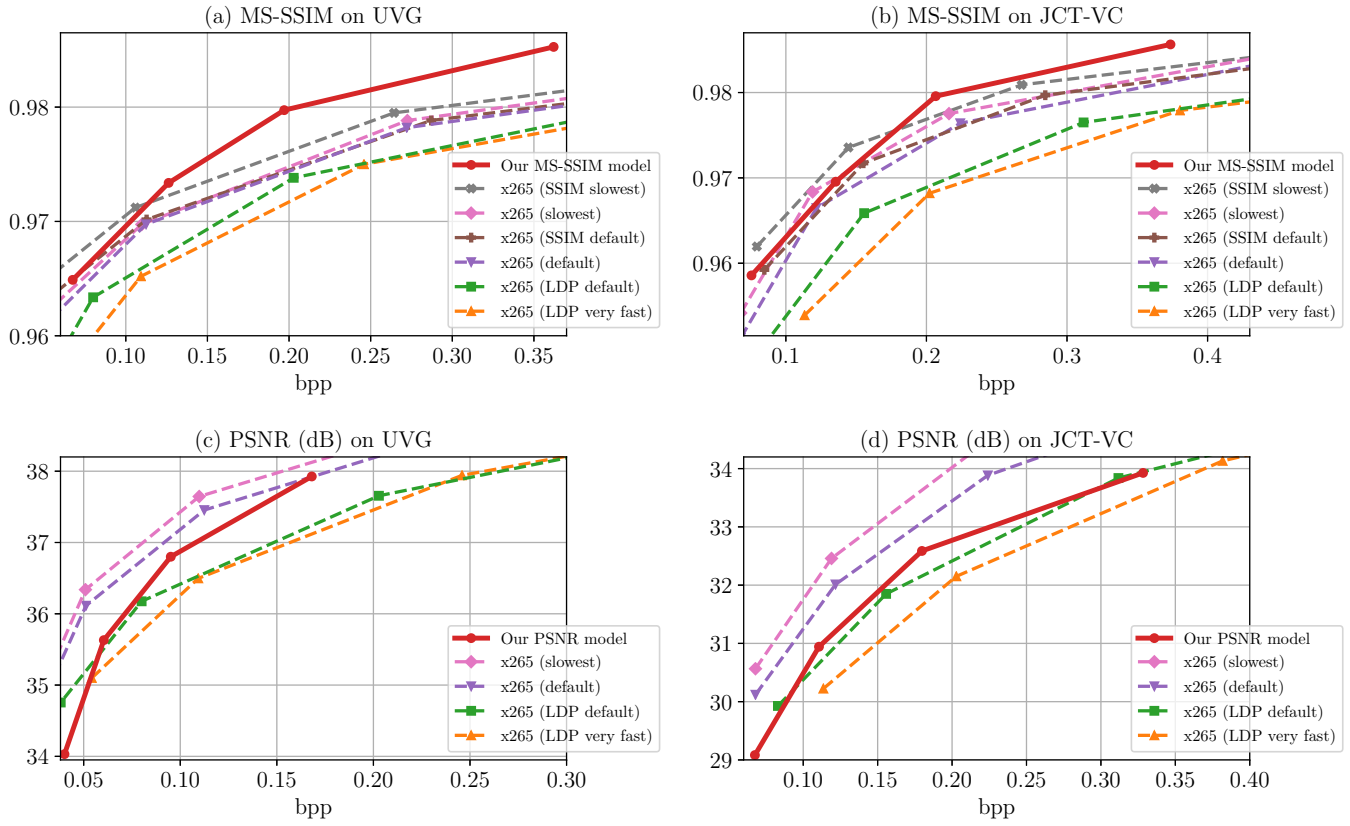


Fig. 7. The rate-distortion performance of our RLVC approach compared with different settings of x265 on the UVG and JCT-VC datasets.

TABLE I
BDBR CALCULATED BY **MS-SSIM** WITH THE ANCHOR OF X265 (LDP VERY FAST). **BOLD** IS THE BEST RESULTS IN LEARNED APPROACHES.

		Learned					Non-learned				
Dataset	Video	DVC [9]	Cheng [10]	Habibian [48]	HLVC [12]	RLVC (Ours)	x265 LDP def.	x265 default	x265 SSIM def.	x265 slowest	x265 SSIM slowest
UVG	*Beauty*	−14.85	-	−44.63	−41.39	**−49.22**	−3.35	3.18	−0.76	6.31	−23.72
	Bosphorus	10.03	-	−13.77	−51.22	**−62.02**	−2.63	−45.35	−48.07	−46.01	−55.27
	HoneyBee	−21.63	-	−4.13	−42.87	**−43.49**	−54.90	−70.78	−67.57	−66.96	−66.58
	Jockey	104.82	-	56.38	6.97	**−12.54**	−13.41	−15.15	−27.32	−20.98	−44.95
	ReadySetGo	2.77	-	89.06	−7.32	**−20.98**	−13.54	−36.94	−40.96	−43.07	−52.11
	ShakeNDry	−20.94	-	−35.10	−32.82	**−40.10**	−24.08	−38.64	−40.96	−45.02	−51.36
	YachtRide	−3.83	-	−21.85	−42.17	**−55.96**	−0.09	−20.76	−23.32	−25.68	−31.69
	Average	8.05	-	3.71	−30.12	**−40.62**	−16.00	−32.06	−35.57	−34.49	−46.52
JCT-VC Class B	*BasketballDrive*	15.47	-	-	−34.98	**−48.10**	2.19	−18.05	−30.26	−22.88	−42.68
	BQTerrace	15.08	-	-	−22.52	**−44.10**	−30.97	−55.70	−50.36	−56.55	−57.09
	Cactus	−21.40	-	-	−43.63	**−53.96**	−26.22	−41.15	−45.28	−45.32	−52.98
	Kimono	−2.67	-	-	−46.79	**−56.73**	−7.24	−13.57	−25.03	−18.63	−34.77
	ParkScene	−20.17	-	-	−39.31	**−49.18**	−9.46	−43.61	−47.04	−48.54	−55.94
	Average	−2.74	-	-	−37.44	**−50.42**	−14.34	−34.42	−39.60	−38.38	−48.69
JCT-VC Class C	*BasketballDrill*	5.54	17.97	-	−18.45	**−32.57**	−18.59	−41.12	−42.70	−46.53	−50.39
	BQMall	4.84	**−38.59**	-	−20.33	−36.88	−12.56	−34.05	−39.04	−43.13	−50.79
	PartyScene	−23.60	−6.53	-	−30.29	**−38.81**	−11.70	−41.53	−42.67	−48.07	−52.30
	RaceHorses (480p)	−14.29	41.07	-	−25.45	**−35.50**	−8.08	−21.69	−22.89	−30.82	−36.61
	Average	−6.88	3.48	-	−23.63	**−35.94**	−12.73	−34.60	−36.82	−42.14	−47.52
JCT-VC Class D	*BasketballPass*	0.67	−44.96	-	−36.24	**−51.40**	−14.45	−32.55	−32.89	−39.03	−42.69
	BlowingBubbles	−29.38	−22.92	-	−39.84	**−49.57**	−11.04	−39.02	−41.12	−45.49	−51.35
	BQSquare	−25.50	−39.60	-	**−97.56**	−44.71	−14.16	−57.31	−56.04	−62.24	−60.37
	RaceHorses (240p)	−19.82	12.60	-	−36.59	**−49.75**	−7.63	−23.15	−24.70	−37.22	−41.15
	Average	−18.51	−23.72	-	**−52.56**	−48.85	−11.82	−38.01	−38.69	−45.99	−48.89
Average on all videos		−2.94	-	-	−35.14	**−43.78**	−14.10	−34.35	−37.45	−39.29	−47.74

In the following, we train our recurrent model in an end-to-end manner on the sequential training frames using loss function of

$$\mathcal{L} = \lambda \cdot \sum_{t=1}^{6} D(\boldsymbol{f}_t, \hat{\boldsymbol{f}}_t) + R_1(\boldsymbol{y}_t^m) + R_1(\boldsymbol{y}_t^r) + \sum_{t=2}^{6} \Big(R_{\text{RPM}}(\boldsymbol{y}_t^m) + R_{\text{RPM}}(\boldsymbol{y}_t^r) \Big). \tag{16}$$

During training, quantization is relaxed by the method in [33] to avoid zero gradients. We follow [12] to set λ as 8, 16, 32 and 64 for MS-SSIM, and as 256, 512, 1024 and 2048 for PSNR. The Adam optimizer [57] is utilized for training. The initial learning rate is set as 10^{-4} for all loss functions (13), (14), (15) and (16). When training the whole model by the final loss of (16), we decrease the learning rate after convergence by the factor of 10 until 10^{-6}.

IV. EXPERIMENTS

A. Settings

The experiments are conducted to validate the effectiveness of our RLVC approach. We evaluate the performance on the same test set as [12], *i.e.*, the JCT-VC [58] (Classes B, C and D) and the UVG [59] datasets. The JCT-VC Class B and UVG are high resolution (1920×1080) datasets, and the JCT-VC Classes C and D are with the resolution of 832×480 and 416×240, respectively. We compare our RLVC approach with the latest learned video compression methods: HLVC [12] (CVPR'20), Liu *et al.* [49] (AAAI'20), Habibian *et al.* [48] (ICCV'19), DVC [9] (CVPR'19), Cheng *et al.* [10] (CVPR'19) and Wu *et al.* [8] (ECCV'18). To compare with the handcrafted video coding standard H.265 [4], we first include the *LDP very fast* setting of x265 into comparison, which is used as the anchor in previous learned compression works [9], [12], [49]. We also compare our approach with the *LDP default*, the *default* and the *slowest* settings of x265. Moreover, the *SSIM-tuned* x265 is also compared with our MS-SSIM model. The detailed configurations of x265 are listed as follows:

- **x265 (LDP very fast):**
  ```
  ffmpeg (input) -c:v libx265
  -preset veryfast -tune zerolatency
  -x265-params "crf=CRF:keyint=10"
  output.mkv
  ```
- **x265 (LDP default):**
  ```
  ffmpeg (input) -c:v libx265
  -tune zerolatency
  -x265-params "crf=CRF" output.mkv
  ```
- **x265 (default):**
  ```
  ffmpeg (input) -c:v libx265
  -x265-params "crf=CRF" output.mkv
  ```

- **x265 (SSIM default):**
  `ffmpeg (input) -c:v libx265 -tune ssim -x265-params "crf=CRF" output.mkv`
- **x265 (slowest):**
  `ffmpeg (input) -c:v libx265 -preset placebo`[1] `-x265-params "crf=CRF" output.mkv`
- **x265 (SSIM slowest):**
  `ffmpeg (input) -c:v libx265 -preset placebo -tune ssim -x265-params "crf=CRF" output.mkv`

In above settings, "`(input)`" is short for "`-pix_fmt yuv420p -s WidthxHeight -r Framerate -i input.yuv`". CRF indicates the compression quality, and lower CRF corresponds to higher quality. We set CRF = 15, 19, 23, 27 for the JCT-VC dataset, and set CRF = 7, 11, 15, 19, 23 for the UVG dataset.

Please refer to the *Supporting Document* for the experimental results on more datasets, such as the conversational video dataset and the MCL-JCV [60] dataset.

B. Performance

Comparison with learned approaches. Fig. 6 illustrates the rate-distortion curves of our RLVC approach in comparison with previous learned video compression approaches on the UVG and JCT-VC datasets. Among the compared approaches, Liu *et al.* [49] and Habibian *et al.* [48] are optimized for MS-SSIM. DVC [9] and Wu *et al.* [8] are optimized for PSNR. HLVC [12] and Agustsson *et al.* [50] train two models for MS-SSIM and PSNR, respectively. As we can see from Fig. 6 (a) and (b), our MS-SSIM model performs competitively to Agustsson *et al.* [50], and outperforms all other learned approaches, including the state-of-the-art MS-SSIM optimized approaches Liu *et al.* [49] (AAAI'20), HLVC [12] (CVPR'20) and Habibian *et al.* [48] (ICCV'19). In terms of PSNR, Fig. 6 (c) and (d) indicate the superior performance of our PSNR model to the PSNR optimized models Agustsson *et al.* [50], HLVC [12] (CVPR'20), DVC [9] (CVPR'19) and Wu *et al.* [8] (ECCV'18). It is worth pointing out that our RLVC approach employs the same motion estimation network as HLVC [12] and DVC [9], and applying the space-scale motion proposed in [50] may further improve the performance of RLVC.

We further tabulate the Bjøntegaard Delta Bit-Rate (BDBR) [61] results calculated by MS-SSIM and PSNR with the anchor of x265 (LDP very fast) in Tables I and II, respectively.[2] Note that, BDBR calculates the average bit-rate difference in comparison with the anchor. Lower BDBR value indicates better performance, and negative BDBR indicates saving bit-rate in comparison with the anchor, *i.e.*, outperforming the anchor. In Tables I and II, the bold numbers are the best results in learned approaches. As Table I shows, in terms of MS-SSIM, the proposed RLVC approach outperforms previous learned approaches on all videos in the high resolution datasets UVG and JCT-VC Class B. In all the 20 test videos, we achieve the best results in learned approaches on 18 videos, and have the best average BDBR performance among all learned approaches. Moreover, Table II shows that, in terms of PSNR, our PSNR model has better performance than all existing learned approaches on all test videos.

TABLE II
BDBR CALCULATED BY **PSNR** WITH THE ANCHOR OF X265 (LDP VERY FAST). **BOLD** IS THE BEST RESULTS IN LEARNED APPROACHES.

	Learned			Non-learned		
Video	DVC [9]	HLVC [12]	RLVC (Ours)	x265 LDP def.	x265 default	x265 slowest
Beauty	−39.63	−48.48	**−56.46**	3.84	4.01	−2.41
Bosphorus	17.57	−23.16	**−35.75**	−4.06	−44.24	−47.72
HoneyBee	24.53	−26.63	**−21.98**	−48.55	−79.03	−80.69
Jockey	90.02	105.21	**82.58**	−9.62	−21.29	−28.96
ReadySetGo	9.03	26.69	**0.03**	−12.68	−39.76	−47.52
ShakeNDry	−25.07	−26.88	**−31.52**	−21.58	−43.43	−50.68
YachtRide	−14.19	−16.34	**−31.30**	−1.95	−19.47	−27.04
Ave. (UVG)	8.89	−1.37	**−13.48**	−13.51	−34.74	−40.72
BasketballDrive	35.24	13.21	**4.40**	−1.92	−20.70	−28.08
BQTerrace	2.28	−4.56	**−20.12**	−28.03	−60.29	−63.44
Cactus	−5.19	−29.09	**−34.71**	−23.66	−48.60	−53.60
Kimono	−10.79	−18.71	**−34.40**	−5.13	−15.41	−22.46
ParkScene	−11.63	−19.59	**−36.16**	−7.73	−45.64	−51.89
Ave. (Class B)	1.98	−11.75	**−24.20**	−13.29	−38.13	−43.89
BasketballDrill	18.03	−3.67	**−11.75**	−21.41	−42.21	−50.16
BQMall	62.28	13.68	**−0.32**	−12.82	−35.31	−45.86
PartyScene	8.61	2.08	**−18.03**	−9.81	−42.35	−50.74
RaceHorses	14.61	19.25	**11.43**	−8.05	−20.53	−30.83
Ave. (Class C)	25.88	7.83	**−4.67**	−13.02	−35.10	−44.40
BasketballPass	42.34	−3.44	**−19.16**	−17.16	−28.73	−37.97
BlowingBubbles	−12.15	−19.19	**−31.67**	−10.96	−38.53	−46.52
BQSquare	22.01	−19.10	**−35.27**	−16.59	−58.64	−68.40
RaceHorses	9.18	−8.55	**−21.93**	−7.90	−22.43	−37.74
Ave. (Class D)	15.34	−12.57	**−27.01**	−13.15	−37.08	−47.66
Ave. (all videos)	11.85	−4.36	**−17.10**	−13.29	−36.13	−43.64

Note that, the latest HLVC [12] (CVPR'20) approach introduces bi-directional prediction, hierarchical structure and post-processing into learned video compression, while the proposed RLVC approach only works in the uni-directional IPPP model without post-processing (as shown in Fig. 2). Nevertheless, our approach still achieves better performance than HLVC [12], validating the effectiveness of our recurrent compression framework with the proposed RAE and RPM networks.

Comparison with x265. The rate-distortion curves compared with different settings of x265 are demonstrated in Fig. 7. As Fig. 7 (a) and (b) show, the proposed MS-SSIM model outperforms x265 (LDP very fast), x265 (LDP default), x265 (default) and x265 (SSIM default) on both the UVG and JCT-VC datasets from low to high bit-rates. Besides, in comparison with the slowest setting of x265, we also achieve better performance on UVG and at high bit-rates on JCT-VC. Moreover, at high bit-rates, we even have higher MS-SSIM performance than the SSIM-tuned slowest setting of x265, which can be seen as the best (MS-)SSIM performance that

[1] *Placebo* is the slowest setting among the 10 speed levels in x265.

[2] Since [8], [49] do not release the result on each video, their BDBR values cannot be obtained.

Fig. 8. The visual results of the MS-SSIM and PSNR models of the proposed RLVC approach in comparison with the default setting of x265.

x265 is able to reach.

Similar conclusion can be obtained from the BDBR results calculated by MS-SSIM in Table I. That is, our RLVC approach averagely reduces 43.78% bit-rate of the anchor x265 (LDP very fast), and outperform x265 (LDP default), x265 (default), x265 (SSIM default) and x265 (slowest). In comparison with x265 (SSIM slowest), we achieve better performance on 8 out of the 20 test videos. We also have better average BDBR result than x265 (SSIM slowest) on JCT-VC Class B, and reach almost the same average performance as x265 (SSIM slowest) on JCT-VC Class D.

In terms of PSNR, Fig. 7 (c) and (d) show that our PSNR model outperforms x265 (LDP very fast) from low to high bit-rates on both the UVG and JCT-VC test sets. Besides, we are superior to x265 (LDP default) at high bit-rates on UVG and in a large of bit-rates on JCT-VC. The BDBR results calculated by PSNR in Table II also indicate that our approach achieves 17.10% less bit-rate than x265 (LDP very fast), and reduces 3.81% more bit-rate than x265 (LDP default). We do not outperform the default and the slowest settings of x265 on PSNR. However, x265 (default) and x265 (slowest) apply advanced strategies in video compression, such as bi-directional prediction and hierarchical frame structure, while our approach only utilizes the uni-directional IPPP mode. Note that, as far as we know, there is no learned video compression approach beats the default setting of x265 in terms of PSNR. The proposed RLVC approach advances the state-of-the-art learned video compression performance and contributes to catching up with the handcrafted standards step by step.

TABLE III
COMPLEXITY (FPS) ON 240P VIDEOS.

	DVC [9]	HLVC [12]	Habibian [48]	RLVC (Ours)
Encoding	23.3	28.8	31.3	15.9
Decoding	39.5	18.3	0.004	32.1

Visual results. The visual results of our MS-SSIM and PSNR models are illustrated in Fig. 8, comparing with the

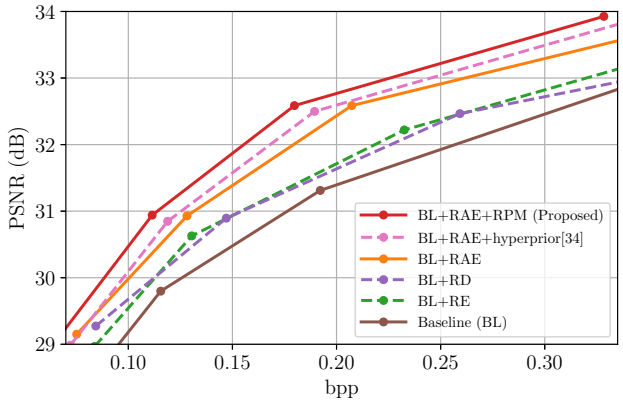


Fig. 9. Ablation results of PSNR (dB) on the JCT-VC dataset.

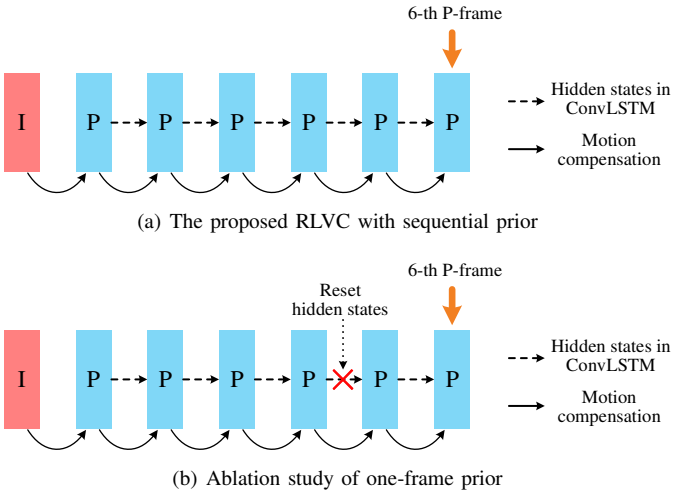


Fig. 10. The ablation study of (a) the proposed sequential prior and (b) one-frame prior.

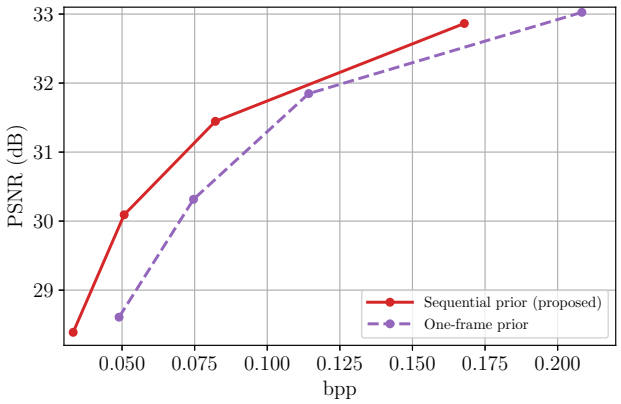


Fig. 11. The average performance of the 6-th P-frame in all GOPs of the JCT-VC dataset. The two curves correspond to Fig. 10 (a) and (b), respectively.

default setting of x265. It can be seen from Fig. 8 that our MS-SSIM model reaches higher MS-SSIM with lower bit-rate than x265, and produces the compressed frame with less blocky artifacts. For our PSNR model, as discussed above, we do not beat the default setting of x265 in terms of PSNR. However, as Fig. 8 shows, our PSNR model also achieves less blocky artifacts and less noise than x265, and is able to reach similar or even higher MS-SSIM than the default setting of x265 in some cases.

Computational complexity. We measure the complexity of the learned approaches on one NVIDIA 1080Ti GPU. The results in terms of frame per second (fps) are shown in Table III. As Table III shows, due to the recurrent cells in our auto-encoders and probability model, the superior performance of our approach is at the cost of the higher encoding complexity than previous approaches. Nevertheless, we have faster decoding than [12], [48], and achieve the real-time decoding on 240p videos with frame rate ≤ 30. Note that, HLVC [12] adopts an enhancement network in the decoder to improve compression quality, which increases decoding complexity. Our RLVC approach (without enhancement) still reaches higher compression performance than HLVC [12], and also has faster decoding speed. Besides, the auto-regressive (PixelCNN-like) probability model used in [48] leads to slow decoding, while the proposed RPM network is more efficient.

C. Ablation studies

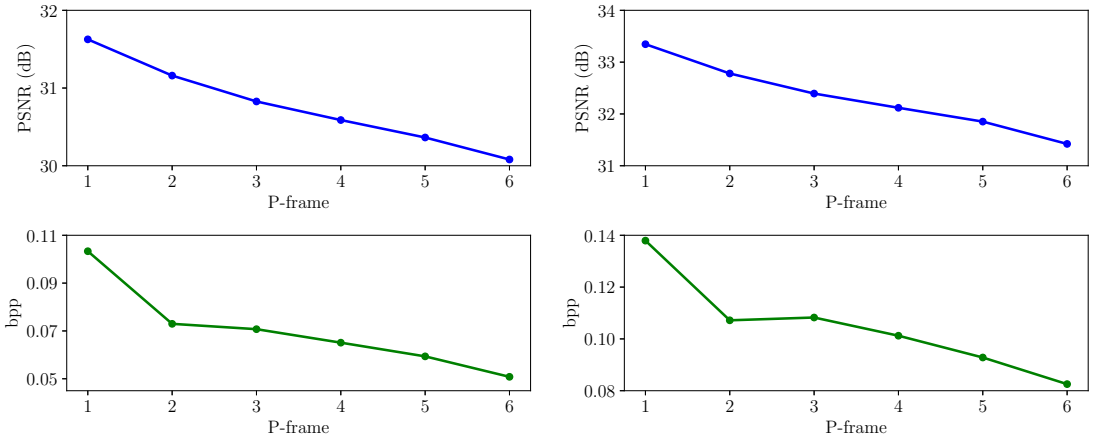
The ablation studies are conducted to verify the effectiveness of each recurrent component in our approach. We define the baseline (BL) as our framework without recurrent cells, *i.e.*, without recurrent cells in auto-encoders and replacing our RPM network with the factorized spatial entropy model [33]. In the following, we enable the recurrent cell in the encoder (BL+RE) and in the decoder (BL+RD), respectively. Then, both of them are enabled, *i.e.*, the proposed RAE network (BL+RAE). Finally, our RPM network is further applied to replace the spatial model [33] (BL+RAE+RPM, *i.e.*, our full model). Besides, we also compare our RPM network with the hyperprior spatial entropy model [34].

The proposed RAE. As Fig. 9 shows, the rate-distortion curves of BL+RE and BL+RD are both above the baseline. This indicates that the recurrent encoder and the recurrent decoder are both able to improve the compression performance. Moreover, combining them together in the proposed RAE, the rate-distortion performance is further improved (shown as BL+RAE). The probable reason is that, because of the dual recurrent cells in both the encoder and decoder, it learns to encode the residual information between the current and previous inputs, which reduces the information content represented by each latent representation, and then the decoder reconstructs the output based on the encoded residual and previous outputs. This results in efficient compression.

The proposed RPM. It can be seen from Fig. 9 that the proposed RPM (BL+RAE+RPM) significantly reduces the bit-rate in comparison with BL+RAE, which uses the spatial entropy model [33]. This proves the fact that at the same compression quality, the temporally *conditional* cross entropy is smaller than the *independent* cross entropy, *i.e.*,

$$\mathbb{E}_{\boldsymbol{y}_t \sim p_t}[-\log_2 q_t(\boldsymbol{y}_t \,|\, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})] < \mathbb{E}_{\boldsymbol{y}_t \sim p}[-\log_2 q(\boldsymbol{y}_t)].$$

Besides, Fig. 9 shows that our RPM network further outperforms the hyperprior spatial entropy model [34], which generates the side information $\boldsymbol{z}_t$ to facilitate the compression



(a) Change of PSNR (dB) and bpp at $\lambda = 512$ (b) Change of PSNR (dB) and bpp at $\lambda = 1024$

Fig. 12. The changes of PSNR (dB) and bpp on consecutive P-frames at (a) $\lambda = 512$ and (b) $\lambda = 1024$.

of $\boldsymbol{y}_t$. This indicates that when compressing video at the same quality, the *temporally* conditional cross entropy is smaller than the *spatial* conditional cross entropy (with the overhead cross entropy of $\boldsymbol{z}_t$), *i.e.*,

$$\mathbb{E}_{\boldsymbol{y}_t \sim p_t}[-\log_2 q_t(\boldsymbol{y}_t \mid \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})] < \mathbb{E}_{\boldsymbol{y}_t \sim p_{\boldsymbol{y}|\boldsymbol{z}}}[-\log_2 q_{\boldsymbol{y}|\boldsymbol{z}}(\boldsymbol{y}_t \mid \boldsymbol{z}_t)] + \mathbb{E}_{\boldsymbol{z}_t \sim p_{\boldsymbol{z}}}[-\log_2 q_{\boldsymbol{z}}(\boldsymbol{z}_t)].$$

The proposed RPM has two benefits over [34]. First, our RPM does not consume overhead bit-rate to compress the prior information, while [34] has to compress $\boldsymbol{z}_t$ into bit stream. Second, our RPM uses the temporal prior of all previous latent representations, while there is only one spatial prior $\boldsymbol{z}_t$ in [34] with much smaller size, *i.e.*, $\frac{1}{16}$ of $\boldsymbol{y}_t$. In conclusion, these studies verify the benefits of applying temporal prior to estimate the conditional probability $q_t(\boldsymbol{y}_t \mid \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})$ in a recurrent manner.

Sequential prior vs. one-frame prior. Fig. 10 (a) shows the recurrent framework of the proposed RLVC, in which the hidden states are transferred through P-frames. This way, the sequential prior can be utilized to compress the upcoming P-frame, *e.g.*, the 6-th P-frame. To show the advantage of applying recurrent cells in RLVC, instead of only using the prior of the intermediately previous frame, we conduct the ablation to reset the hidden states to the initial state before the 5-th P-frame (as shown in Fig. 10 (b)) and analyze the decrease of compression performance on the 6-th frame. Fig. 11 illustrates the performance of the 6-th frame in all Group of Pictures (GOPs) of all videos in the JCT-VC test set. It can be seen from Fig. 11 that only utilizing the one-frame prior, *i.e.*, resetting states before the 5-th P-frame, obviously decreases the rate-distortion performance of the 6-th P-frame, in comparison with using the sequential prior in our RLVC approach. This validates the effectiveness and advantage of the recurrent framework of RLVC.

Change of PSNR and bpp along consecutive P-frames. Fig. 12 shows the change of PSNR (dB) and bit-rate along sequential P-frames after the I-frame. The results are averaged among all GOPs in the JCT-VC test set. It can be seen from Fig. 12 that both PSNR and bit-rate decrease when the distance from I-frame increases. This is probably because of the combined effect of the richer prior transferred in recurrent networks and the farther distance from I-frame. In terms of PSNR, the decrease of quality on the frame which is used to predict the next frame by motion compensation (refer to Fig. 2) may lead to quality decrease of the next frame. However, given more previous frames, the proposed Recurrent Auto-Encoder (RAE) and Recurrent Probability Model (RPM) are both with richer temporal prior. Therefore, the RAE learns to generate more efficient latent representation and the RPM learns to more accurately model the conditional probability function. This way, the bit-rate drops along sequential P-frames.

D. Combining RPM with spatial probability models

It is worth pointing out that the proposed RPM network is flexible to be combined with various spatial probability models, *e.g.*, [6], [7], [34]. As an example, we train a model combining the proposed approach with the hyperprior spatial probability model [34], which is illustrated in Fig. 13. This combined model only slightly improves our approach, *i.e.*, 0.36% bit-rate reduction on the JCT-VC dataset. On the one hand, such slight improvement indicates that due to the high correlation among video frames, the previous latent representations are able to provide most of the useful information, and the spatial prior, which leads to bit-rate overhead, is not very helpful to further improve the performance. This validates the effectiveness of our RPM network. On the other hand, it also shows the flexibility of our RPM network to combine with spatial probability models, *e.g.*, replacing the spatial model in Fig. 13 with [34], [6] or [7][3], and the possibility to further advance the performance.

V. Discussion

A. GOP structure

In this section, we first discuss the performance of our approach when compressing video with different Group of

[3]Since [6], [7] do not release the training codes, we are not able to learn the model combining RPM with [6], [7].

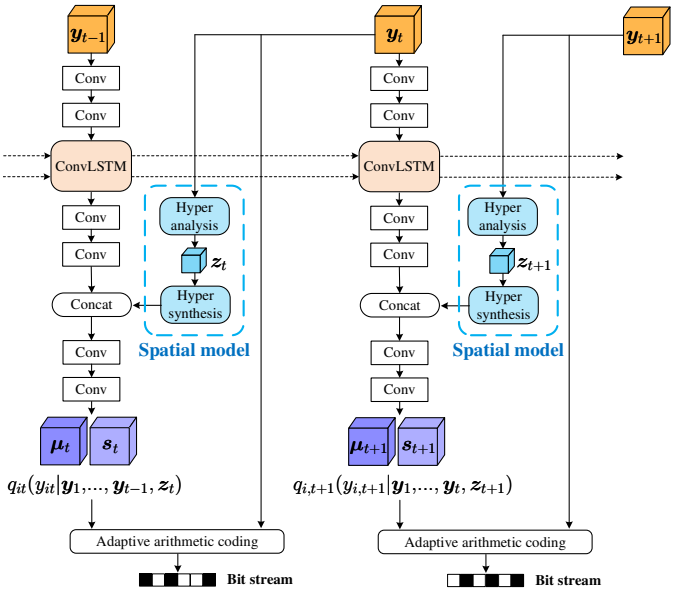
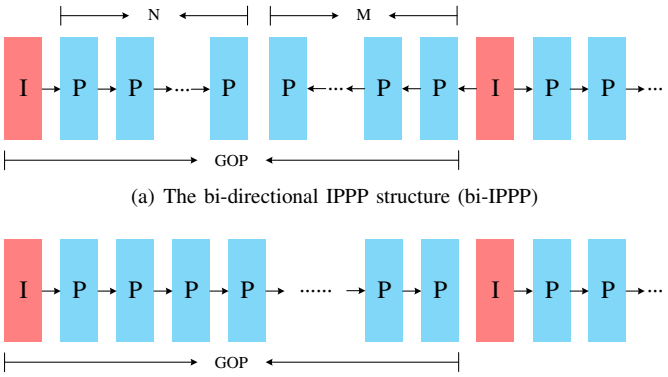


Fig. 13. The probability model combining the proposed RPM with the spatial hyperprior model [34].

Picture (GOP) structures. Fig. 14 shows two kinds of GOP structures which are possible to be used for the proposed RLVC approach. Fig. 14 (a) demonstrates a bi-directional IPPP (bi-IPPP) structure, which reduces the longest distance between I- and P-frames to suppress error propagation. Specifically, N P-frames after the previous I-frame are compressed recurrently by RLVC, then the next I-frame is compressed and the M P-frames before it are recurrently compressed in the reverse direction. This way, the GOP size equals to $N+M+1$. Fig. 14 (b) shows the normal IPPP structure (uni-IPPP), which compressed frames in the natural order.

Fig. 15 illustrates the rate-distortion curves of our RLVC approach with various GOP structures on the JCT-VC dataset. In Fig. 15, we first show the performance of GOP $=$ 13 with bi-IPPP mode using $N = M = 6$. It achieves the best performance, and this structure is used in the experiments in Section IV. Moreover, Fig. 15 shows that when enlarging the GOP size to 20 (bi-IPPP), the performance is still competitive with GOP $=$ 13 (bi-IPPP) with only slight degradation. Then, we also analyze the uni-IPPP mode (dash lines) from small to large GOP sizes, *i.e.*, GOP $= 10, 13$ and 20. It can be seen from Fig. 15 that the performance of the uni-IPPP mode are very similar among different GOP sizes, which are lower than the bi-IPPP mode by around 0.3 dB to 0.5 dB. Note that it also happens to other traditional and learned video compression approaches that bi-directional prediction achieves better performance than the uni-directional prediction. Also, the bi-directional prediction is utilized in previous learned video compression approaches, *e.g.*, Wu *et al.* [8] and HLVC [12], and we outperform [8], [12] as the results shown in Fig. 6.

In conclusion, the proposed RLVC approach is compatible to various GOP sizes, and especially adjustable to GOP $=$ 20 (bi-IPPP) without obvious degradation of compression performance.



(a) The bi-directional IPPP structure (bi-IPPP)

(b) The uni-directional IPPP structure (uni-IPPP)

Fig. 14. Two kinds of GOP structures for our RLVC approach.

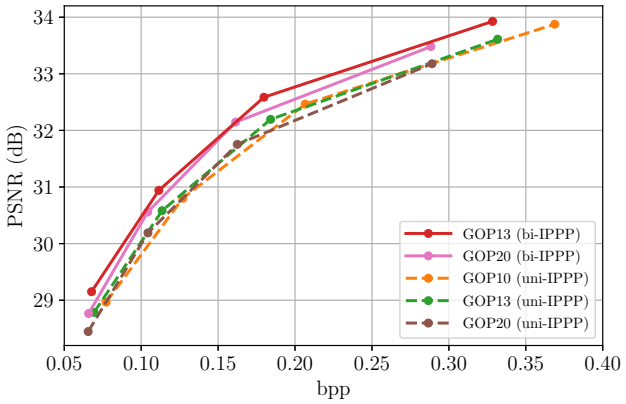


Fig. 15. The performance of different GOP structures on the JCT-VC dataset.

B. Error propagation

Learned video compression approaches usually suffer from error propagation when the distance between I- and P-frames increases. We analyzed the rate-distortion curves of 12 consecutive P-frames compressed by the proposed RLVC model. Note that the length of 12 is twice as long as the training samples (6 P-frames plus an I-frame). Fig. 16 shows the average performance of different frames with the GOP = 13 (uni-IPPP) setting. The rate-distortion curves are averaged among all GOPs in the JCT-VC dataset. In Fig. 16, frame 7 (the 6-th P-frame) and its previous frames are within the training length of our models. It can be seen that after going out of the training length, frames 11 and 13 indeed have lower performance than frame 7, *e.g.*, the PSNR drops around 0.5 dB from frame 7 to frame 13. This indicates that error propagation also exists in the proposed RLVC approach. However, the performance on frame 13 is even better than frame 11 at low bit-rates and maintains comparable performance with frame 11 at high bit-rates. A similar phenomenon can be observed from frame 3 to frame 7, *i.e.*, frame 7 achieves higher performance at low bit-rates than frame 3. This is probably because the proposed recurrent compression network contains richer temporal prior when moving forwards frame-by-frame, thus facilitating the compression of the frames which are farther from I-frame. To a certain degree, this mechanism is able to combat

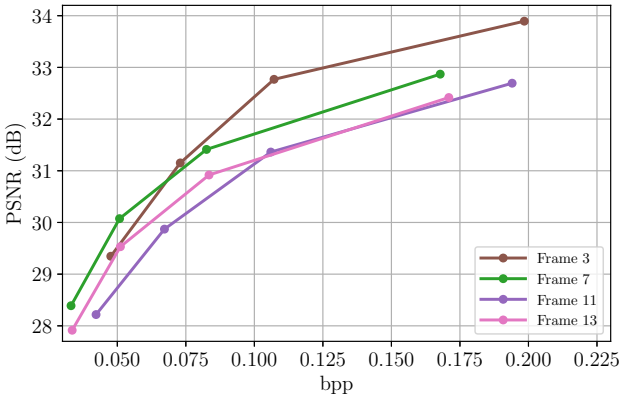


Fig. 16. The average performance of different P-frames among all GOPs in the JCT-VC dataset.

against the error propagation caused by the increasing distance from I-frame. Therefore, as Fig. 15 shows, the compression performance does not degrade obviously when enlarging the GOP size.

VI. Conclusion and Future Work

This paper has proposed a recurrent learned video compression approach. Specifically, we proposed recurrent auto-encoders to compress motion and residual, fully exploring the temporal correlation in video frames. Then, we showed how modeling the conditional probability in a recurrent manner improves the coding efficiency. The proposed recurrent auto-encoders and recurrent probability model significantly expands the range of reference frames, which has not been achieved in previous learned *as well as* handcrafted standards. The experiments validate that the proposed approach outperforms all previous learned approaches and the LDP default setting of x265 in terms of both PSNR and MS-SSIM, and also outperforms x265 (slowest) on MS-SSIM. The ablation studies verify the effectiveness of each recurrent component in our RLVC approach, and show the flexibility of the proposed RPM network to combine with spatial probability models.

Moreover, the proposed method can inspire traditional codecs, particularly the methods that integrate deep networks in traditional codecs, to adopt recurrent networks to improve their performance. For instance, Liu *et al.* [41] and Choi *et al.* [42] improves the motion compensation of HEVC by utilizing single deep network on each frame, and Li *et al.* [44], [45] replace the in-loop of HEVC by non-recurrent deep networks. These methods are possible to be advanced by employing the recurrent networks (similar to the proposed approach) to further improve the traditional codecs, such as HEVC.

In this paper, the recurrent framework of the proposed approach still relies on the warping operation and motion compensation to reduce the temporal redundancy. Therefore, it is a promising future work to eliminate the dependency on optical-flow-based motion detection, and learn a fully recurrent network or adopt an attention mechanism (*e.g.*, transformer [62] based) for learned video compression. Besides, the proposed approach achieves superior performance at the cost of higher encoding complexity. Another future work is to study reducing complexity and the trade-off between complexity and rate-distortion performance. For example, the proposed network may be sped up by reducing the layer number and channel numbers in the auto-encoders and the motion compensation network, or by utilizing a more time-efficient optical flow network for motion prediction.

References

[1] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2017-2022 white paper," https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html.

[2] D. J. Le Gall, "The MPEG video compression algorithm," *Signal Processing: Image Communication*, vol. 4, no. 2, pp. 129–140, 1992.

[3] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 560–576, 2003.

[4] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[5] D. Minnen, J. Ballé, and G. D. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 10 771–10 780.

[6] J. Lee, S. Cho, and S.-K. Beack, "Context-adaptive entropy model for end-to-end optimized image compression," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[7] Y. Hu, W. Yang, and J. Liu, "Coarse-to-fine hyper-prior modeling for learned image compression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[8] C.-Y. Wu, N. Singhal, and P. Krahenbuhl, "Video compression through image interpolation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 416–431.

[9] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao, "DVC: An end-to-end deep video compression framework," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 11 006–11 015.

[10] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Learning image and video compression through spatial-temporal energy compaction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10 071–10 080.

[11] A. Djelouah, J. Campos, S. Schaub-Meyer, and C. Schroers, "Neural inter-frame compression for video coding," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 6421–6429.

[12] R. Yang, F. Mentzer, L. Van Gool, and R. Timofte, "Learning for video compression with hierarchical quality and recurrent enhancement," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[13] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length and helmholtz free energy," in *Advances in neural information processing systems*, 1994, pp. 3–10.

[14] K. Cho, "Simple sparsification improves sparse denoising autoencoders in denoising highly corrupted images," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2013, pp. 432–440.

[15] L. Gondara, "Medical image denoising using convolutional denoising autoencoders," in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2016, pp. 241–246.

[16] K. G. Lore, A. Akintayo, and S. Sarkar, "Llnet: A deep autoencoder approach to natural low-light image enhancement," *Pattern Recognition*, vol. 61, pp. 650–662, 2017.

[17] S. Park, S. Yu, M. Kim, K. Park, and J. Paik, "Dual autoencoder network for retinex-based low-light image enhancement," *IEEE Access*, vol. 6, pp. 22 084–22 093, 2018.

[18] K. Zeng, J. Yu, R. Wang, C. Li, and D. Tao, "Coupled deep autoencoder for single image super-resolution," *IEEE transactions on cybernetics*, vol. 47, no. 1, pp. 27–37, 2015.

[19] R. Wang and D. Tao, "Non-local auto-encoder with collaborative stabilization for image restoration," *IEEE Transactions on Image Processing*, vol. 25, no. 5, pp. 2117–2129, 2016.

[20] A. Karpathy, J. Johnson, and L. Fei-Fei, "Visualizing and understanding recurrent networks," 2016.

[21] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.

[22] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," *arXiv preprint arXiv:1602.02410*, 2016.

[23] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 2625–2634.

[24] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

[25] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[26] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2015, pp. 3156–3164.

[27] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using LSTMs," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2015, pp. 843–852.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[29] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar, "Variable rate image compression with recurrent neural networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[30] G. Toderici, D. Vincent, N. Johnston, S. Jin Hwang, D. Minnen, J. Shor, and M. Covell, "Full resolution image compression with recurrent neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5306–5314.

[31] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. V. Gool, "Soft-to-hard vector quantization for end-to-end learning compressible representations," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 1141–1151.

[32] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[33] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[34] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[35] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool, "Conditional probability models for deep image compression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4394–4402.

[36] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang, "Learning convolutional networks for content-weighted image compression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 3214–3223.

[37] N. Johnston, D. Vincent, D. Minnen, M. Covell, S. Singh, T. Chinen, S. Jin Hwang, J. Shor, and G. Toderici, "Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4385–4393.

[38] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG 2000 still image compression standard," *IEEE Signal processing magazine*, vol. 18, no. 5, pp. 36–58, 2001.

[39] F. Bellard, "BPG image format," https://bellard.org/bpg/.

[40] M. Xu, T. Li, Z. Wang, X. Deng, R. Yang, and Z. Guan, "Reducing complexity of HEVC: A deep learning approach," *IEEE Transactions on Image Processing*, vol. 27, no. 10, pp. 5044–5059, 2018.

[41] J. Liu, S. Xia, W. Yang, M. Li, and D. Liu, "One-for-all: Grouped variation network-based fractional interpolation in video coding," *IEEE Transactions on Image Processing*, vol. 28, no. 5, pp. 2140–2151, 2018.

[42] H. Choi and I. V. Bajić, "Deep frame prediction for video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[43] Y. Dai, D. Liu, and F. Wu, "A convolutional neural network approach for post-processing in HEVC intra coding," in *Proceedings of the International Conference on Multimedia Modeling (MMM)*. Springer, 2017, pp. 28–39.

[44] T. Li, M. Xu, R. Yang, and X. Tao, "A DenseNet based approach for multi-frame in-loop filter in HEVC," in *Proceedings of the Data Compression Conference (DCC)*. IEEE, 2019, pp. 270–279.

[45] T. Li, M. Xu, C. Zhu, R. Yang, Z. Wang, and Z. Guan, "A deep learning approach for multi-frame in-loop filter of HEVC," *IEEE Transactions on Image Processing*, 2019.

[46] T. Chen, H. Liu, Q. Shen, T. Yue, X. Cao, and Z. Ma, "DeepCoder: A deep neural network based video compression," in *Proceedings of the IEEE Visual Communications and Image Processing (VCIP)*. IEEE, 2017, pp. 1–4.

[47] Z. Chen, T. He, X. Jin, and F. Wu, "Learning for video compression," *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[48] A. Habibian, T. van Rozendaal, J. M. Tomczak, and T. S. Cohen, "Video compression with rate-distortion autoencoders," in *Proceedings of the IEEE International Conference of Computer Vision (ICCV)*, 2019.

[49] H. Liu, L. Huang, M. Lu, T. Chen, and Z. Ma, "Learned video compression via joint spatial-temporal correlation exploration," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[50] E. Agustsson, D. Minnen, N. Johnston, J. Balle, S. J. Hwang, and G. Toderici, "Scale-space flow for end-to-end optimized video compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 8503–8512.

[51] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves *et al.*, "Conditional image generation with pixelcnn decoders," in *Advances in neural information processing systems*, 2016, pp. 4790–4798.

[52] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4161–4170.

[53] G. G. Langdon, "An introduction to arithmetic coding," *IBM Journal of Research and Development*, vol. 28, no. 2, pp. 135–149, 1984.

[54] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[55] N. Balakrishnan, *Handbook of the logistic distribution*. CRC Press, 1991.

[56] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video enhancement with task-oriented flow," *International Journal of Computer Vision*, vol. 127, no. 8, pp. 1106–1125, 2019.

[57] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[58] F. Bossen, "Common test conditions and software reference configurations," *JCTVC-L1100*, vol. 12, 2013.

[59] A. Mercat, M. Viitanen, and J. Vanne, "UVG dataset: 50/120fps 4K sequences for video codec analysis and development," in *Proceedings of the 11th ACM Multimedia Systems Conference*, 2020, pp. 297–302.

[60] H. Wang, W. Gan, S. Hu, J. Y. Lin, L. Jin, L. Song, P. Wang, I. Katsavounidis, A. Aaron, and C.-C. J. Kuo, "MCL-JCV: a JND-based H.264/AVC video quality assessment dataset," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 1509–1513.

[61] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," *VCEG-M33*, 2001.

[62] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008.

Ren Yang is a doctoral student at ETH Zurich, Switzerland. He received M.Sc. degree in 2019 at the School of Electronic and Information Engineering, Beihang University, China, and obtained the B.Sc. degree at Beihang University in 2016. His research interests mainly include computer vision and video compression. He has published several papers in top international journals and conference proceedings, such as IEEE TPAMI, IEEE TIP, IEEE TCSVT, CVPR, ICCV and ICME. He serves as a reviewer for top conferences and journals, such as ECCV, IEEE TIP, IEEE TCSVT, IEEE TMM and Elsevier's SPIC and NEUCOM. He is the winner of the Three Minute Competition at IEEE ICME 2019.

Fabian Mentzer is a doctoral student of the Computer Vision Laboratory (CVL) at ETH Zurich, Switzerland. He received his M.Sc. degree in Electrical Engineering and Information Technology from ETH Zurich in 2016, and received his B.Sc. degree from ETH Zurich in 2014. His current research interests include deep learning, learned lossy and lossless image compression, and learned video compression. He has published several papers in top international conferences, such as CVPR, ICCV and NeurIPS. He regularly serves as a reviewer for top conferences such as CVPR, ICCV, ECCV, and NeurIPS. He is a co-organizer of the CLIC workshop at CVPR.

Luc Van Gool received the degree in electromechanical engineering at the Katholieke Universiteit Leuven, in 1981. Currently, he is a professor at the Katholieke Universiteit Leuven in Belgium and the ETH in Zurich, Switzerland. He leads computer vision research at both places, where he also teaches computer vision. He has been a program committee member of several major computer vision conferences. His main interests include 3D reconstruction and modeling, object recognition, tracking, and gesture analysis, and the combination of those. He received several Best Paper awards and was nominated Distinguished Researcher by the IEEE Computer Science committee. He is a co-founder of 10 spin-off companies. He is a member of the IEEE.

Radu Timofte received his PhD degree in Electrical Engineering from the KU Leuven, in 2013. He is currently a lecturer and a group leader at ETH Zurich, Switzerland. He is a member of the editorial board of top journals such as IEEE TPAMI, Elsevier's CVIU and NEUCOM, and SIAM's SIIMS. He regularly serves as a reviewer and as an area chair for top conferences such as CVPR, ICCV, IJCAI, and ECCV. His work received several awards. He is a co-founder of Merantix and a co-organizer of NTIRE, CLIC, AIM, and PIRM workshops and challenges. His current research interests include deep learning, implicit models, compression, image restoration and enhancement.

Learning for Video Compression with Recurrent Auto-Encoder and Recurrent Probability Model

– Supporting Document –

A. Performance on conversational video

To validate the generalization ability of the proposed approach, we test our approach on JCT-VC Class E, which is a conversational video dataset. It can be seen from Fig. 17 (a) and (b) that our RLVC approach outperforms the learned approaches DVC [9] and Liu *et al.* [49] in terms of both MS-SSIM and PSNR.[4] Fig. 17 (c) shows that our MS-SSIM model outperforms x265 (LDP default) and x265 (LDP very fast) for all bit-rates. We further outperform all other settings (including the SSIM-tuned slowest setting) of x265 at medium and high bit-rates in terms of MS-SSIM, and we are comparable with them at low bit-rates. In terms of PSNR, Fig. 17 (d) shows that our PSNR model is better than x265 (LDP veryfast), and outperforms x265 (LDP default) when bpp $>$ 0.05. The same as on UVG and JCT-VC Classes B, C and D, we do not outperform x265 (default) and x265 (slowest) on JCT-VC Class E in terms of PSNR. Recall that, x265 (default) and x265 (slowest) use bi-directional prediction and hierarchical frame structure, but only the uni-directional IPPP mode is applied in our approach.

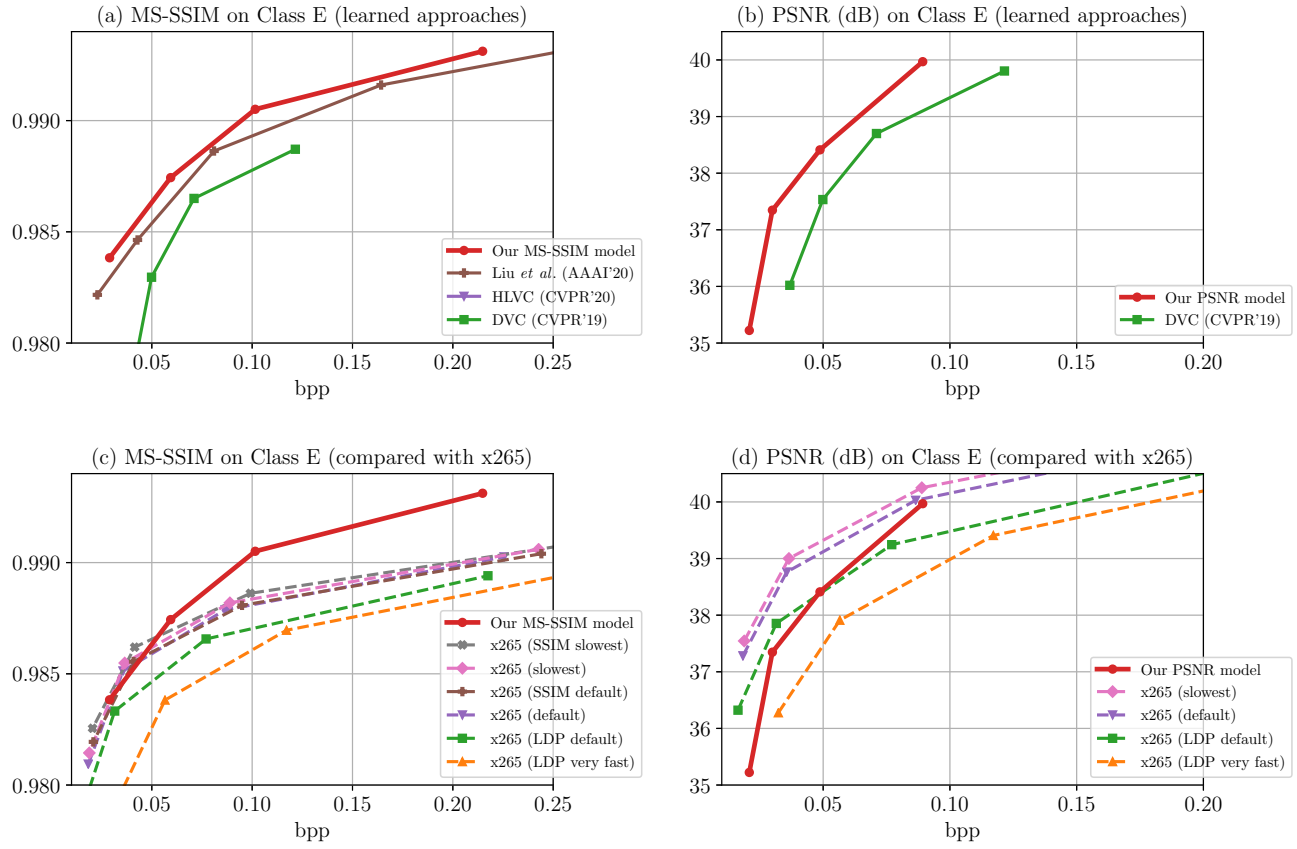


Fig. 17. The rate-distortion performance on JCT-VC Class E in comparison with learned approaches and the different settings of x265.

B. Performance on the MCL-JCV dataset

Fig. 18 demonstrates the rate-distortion performance on the MCL-JCV dataset[5], which contains 30 videos with the resolution of 1920×1080. As Fig. 18 shows, the proposed MS-SSIM model outperforms the LDP default and the LDP very fast settings of x265, and also outperforms x265 (default), x265 (SSIM default), x265 (slowest) and x265 (SSIM slowest) at high bit-rates.

[4]Other learned approaches are not tested on JCT-VC Class E, and the MS-SSIM optimized approach Liu *et al.* [49] does not have results on PSNR.
[5]The MCL-JCV dataset is available at http://mcl.usc.edu/mcl-jcv-dataset/.

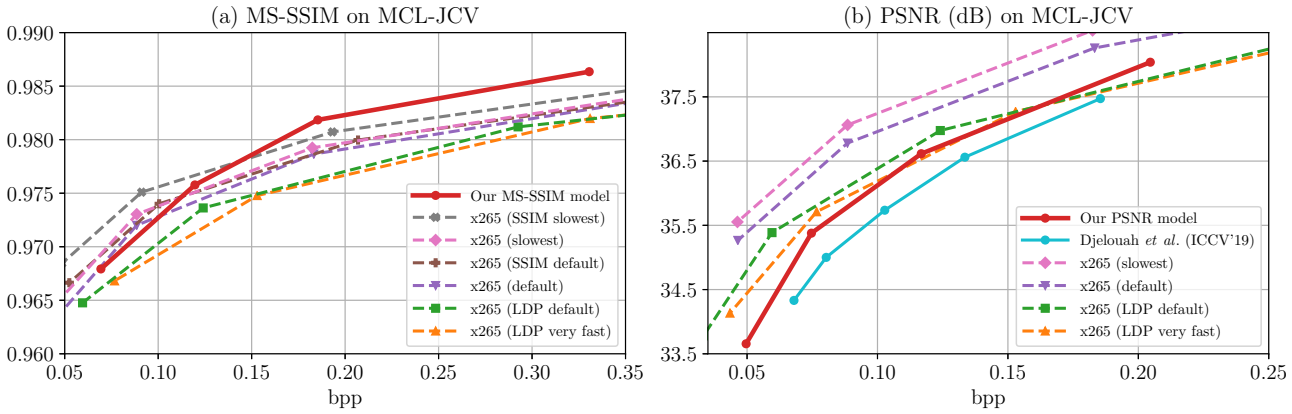


Fig. 18. The rate-distortion performance on the MCL-JCV dataset. The learned approaches are shown in solid lines and x265 is shown in dash lines.

Our MS-SSIM model is comparable with x265 (default) at low bit-rates in terms of MS-SSIM. In terms of PSNR, the proposed PSNR model achieves better performance than the learned video compression approach Djelouah *et al.* [11] (ICCV'19). Note that Djelouah *et al.* [11] compresses video frame with bi-directional prediction, while the proposed approach only works in the IPPP mode. This proves the superior performance of the proposed recurrent video compression approach. Fig. 18 also indicates that we are comparable with x265 (LDP very fast) on PSNR when bpp $>$ 0.1, and even better than x265 (LDP default) at bpp = 0.2. The same as on other datasets, out PSNR model do not reach better performance than x265 (default) and x265 (slowest), which adopts complicated frame structure and coding strategies.