High-quality Single-model Deep Video Compression with Frame-Conv3D and Multi-frame Differential Modulation

Wenyu Sun[0000−0002−4793−0972], Chen Tang, Weigui Li, Zhuqing Yuan, Huazhong Yang[0000−0003−2421−353X], and Yongpan Liu[0000−0002−4892−2309]

Tsinghua University, Beijing, China
ypliu@tsinghua.edu.cn

Abstract. Deep learning (DL) methods have revolutionized the paradigm of computer vision tasks and DL-based video compression is becoming a hot topic. This paper proposes a deep video compression method to simultaneously encode multiple frames with Frame-Conv3D and differential modulation. We first adopt Frame-Conv3D instead of traditional Channel-Conv3D for efficient multi-frame fusion. When generating the binary representation, the multi-frame differential modulation is utilized to alleviate the effect of quantization noise. By analyzing the forward and backward computing flow of the modulator, we identify that this technique can make full use of past frames' information to remove the redundancy between multiple frames, thus achieves better performance. A dropout scheme combined with the differential modulator is proposed to enable bit rate optimization within a single model. Experimental results show that the proposed approach outperforms the H.264 and H.265 codecs in the region of low bit rate. Compared with recent DL-based methods, our model also achieves competitive performance.

1 Introduction

Since video data have contributed to more than 80% of internet traffic [8], video is playing an increasingly important role in human life. Therefore, an efficient video compression system is in highly demand. In the past decades, standard video compression algorithms require large amounts of hand-crafted modules. While they have been well engineered and thoroughly tuned for each local module, they cannot be end-to-end optimized for emerging video applications such as object detection, VR applications, video understanding, and so on. Recently, deep learning (DL) methods achieve great success in various computer vision tasks. DL-based image and video compression approaches have achieved comparable or even better performance than the traditional codecs [4, 6, 14, 21, 22, 26]. Inspired by recent advance in DL-based image and video compression works, we propose a fully end-to-end deep video compression model. The keynotes of this paper are summarized as follows.

Frame 3D convolution for efficient multi-frame fusion. 3D convolution (Conv3D) has been proved as an efficient module for video processing [7, 10, 13].

Traditional Conv3D slides a window along three dimensions: depth, height and width. Then the partial results are added up along channels. For video tasks, the depth dimension is commonly along multiple frames and for clarity we denote this traditional way as Channel-Conv3D. Although Channel-Conv3D can extract spatial and temporal information of multiple frames at the same time, it needs deeper network to slide the window through all input frames, thus causes high computing cost. To resolve this, we introduce Frame-Conv3D which sets the channel as depth and adds up the partial result along frames. This operation can learn the temporal features of all input frames within one convolution layer.

Multi-frame differential modulation to alleviate the effect of quantization noise in training. Despite various block structures, most works follow an auto-encoder framework with an extra quantizer after the middle bottle layer [14, 17, 21, 22, 26]. The quantizer is necessary to achieve the binary representation and generate the bit-streams. However, it complicates gradient based learning since it is inherently non-differentiable. To make the quantizer trainable, one way is to use soft assignment to approximate quantization [4] and another is to model it with additive uniform noise [21]. Although these methods can enable propagating the gradients, the binarization is in pixel-level and the similarity of frames is not adopted, thus introduce much quantization noise. In this paper, we propose the multi-frame differential modulation that binarize the residual between multiple frames. This method can effectively remove the redundant information between frames and thus minimize the effects of quantization noise during training.

Bit rate optimization within single model. To enable bit rate optimization, we propose a dropout scheme to optimize different bit-rate levels within a single model. Our approach is dropping some of the binary representations for transmitting bit. This is an important setting to save computing cost and transmitting bandwidth when variable video qualities are required.

We compare the proposed model with state-of-art video compression codecs (H.264, H.265) and recent DL-based methods [14, 26] on two standard datasets of uncompressed video: UVG [2] and HEVC Standard Test Sequences [19]. Our video codec outperforms the standard H.264, H.265 and DL-based method [26] in compression quality measured by peak signal-to-noise ratio (PSNR) and multiscale structural similarity (MS-SSIM) [24]. It also on par with the deep codecs of [14].

The rest of this paper is organized as follows. Section 2 reviews the related works in DL-based image and video compression. Section 3 describes the proposed framework and illustrates the detailed information in implementation. Section 4 gives the experimental results to show the efficiency of the proposed method, followed with a conclusion in section 5.

2 Related Work

2.1 DL-based Image Compression

DL-based image compressing methods have attracted extensive attention in recent years. Auto-encoders [3–6, 12, 15, 16, 20] and recurrent neural networks (RNNs) [21, 22] are two popular architectures in image compression. Common loss functions for optimizing the network are the mean-squared error (MSE) [4, 5, 20] and MS-SSIM [6, 22]. To improve the subjective visual quality, generative adversarial networks (GANs) are adopted in [3, 16]. In addition, other techniques to improve the image compression performance includes generalized divisive normalization (GDN) [5], multi-scale image decomposition [16], and importance map [12]. These learned approaches provide well guidance for video compression.

2.2 DL-based Video Compression

Compared with image compression, video needs efficient methods to remove the inter-picture redundancy. For this, spatio-temporal auto-encoder is an effective structure, as it extend the convolutional auto-encoder formulation with the ability to extract features of spatial and temporal information at the same time. Chen *et al.* [23] divide video frames into 32×32 blocks and use an anto-encoder to compress the block. They perform motion estimation and compensation with traditional methods and the encoded representations are directly quantized and coded by the Huffman method. This approach is not totally end-to-end and cannot be competitive with H.264. Wu *et al.* [26] propose image interpolation and Conv-LSTM to compress frames iteratively. To reconstruct high-quality video, the Conv-LSTM based codec has to work for several recurrent loops, which results in long running times for both encoding and decoding. Lu *et al.* [14] propose a real end-to-end deep video coding scheme. They employ an extra motion compensation network to calculate the optical flow and compensate for motion between current and previous frames. This scheme achieves better compression efficiency than H.264, and can be competitive with H.265 when evaluated with MS-SSIM. However, multiple models have to be trained in [14] for different levels of bit rate.

2.3 Quantizer for Deep Learning

To generate the bit-stream for image or video compression, a trainable quantizer is necessary in DL framework. Binarizing feature maps in neural networks has been studied in several works for network quantization [9] and image compression [4, 21, 22]. To propagate gradients through the non-differentiable quantizer, methods can be divided into two categories: stochastic regularization [9, 22] and soft assignment [4]. The former replaces quantization by adding noise and the later adopts a soft function to approximate the $\text{sign}(x)$. Although these quantizers can be plugged in the DL framework for backward optimization, the methods are pixel independent. They binarize each pixel without considering the

dependency between nearing pixels. As a result, the mentioned quantizer cannot benefit from frame similarity to compress the features into binary. Instead, this paper propose a quantizer based on differential modulation, which can effectively eliminating redundant information of multiple frames for binary compression.

3 Proposed Method

3.1 Overview of the Proposed Method

A video is composed by a sequence of N frames: $F^N = \{f_1, f_2, ..., f_N\}$, where each frame $f_i \in R^{C\times H\times W}$ has $H \times W$ pixels and C channels (for RGB frames, C is 3). If we stack these N frames in the second dimension together, the input video sequence can also be described as a 4D tensor: $F^N \in R^{C\times N\times H\times W}$ with spatial and temporal information. Fig. 1 gives a high-level description of the proposed video compressing framework. We encode and decode multiple frames together to improve the compressing efficiency. As an auto-encoder, the network structure is composed by an encoder E_ϕ, a differential quantizer Q and a decoder D_θ, where ϕ and θ are the parameters of neural network. Assuming N frames $\hat{F}^N_{t-1}$ have been decoded at last time $t-1$, to encode the N frames F^N_t at current time t, we first apply a prediction network P_σ on $\hat{F}^N_{t-1}$ and then encode the residual $r_t = F^N_t - P_\sigma(\hat{F}^N_{t-1})$, where σ is the parameter of the prediction network. The decoder will reconstruct the residual $\hat{r}_t$ as side information. Algorithm flow for time step t is shown as follows.

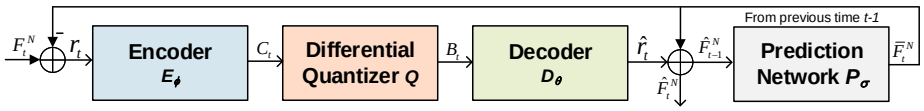


Fig. 1: Proposed end-to-end video compression network composed by the prediction network P_σ, the encoder E_ϕ, the quantizer with differential modulation, and the decoder D_θ.

Input: previous decoded N frames $\hat{F}^N_{t-1} \in R^{3\times N\times H\times W}$ and current N frames $F^N_t \in R^{3\times N\times H\times W}$ to encode.

Step 1: prediction network. Based on previous decoded frames $\hat{F}^N_{t-1}$, we use a 3D resnet model P_σ in Fig. 2(d) to predict the current frames $\bar{F}^N_t = P_\sigma(\hat{F}^N_{t-1})$. 3D resnet block (ResB3D) in Fig. 2(a) has been proved effective for many video tasks such as classification and super-resolution [7, 10, 13].

Step 2: encoder. Based on the predicted frames $\bar{F}^N_t$, we encode the residual $r_t = F^N_t - \bar{F}^N_t$. The encoder E_ϕ is composed by four parts, which is shown in Fig. 2(b). First, we up-sample the channel number from 3 to 64 by ChannelConv3D. Then a stack of ResB3D with max-pooling layers in height and width are introduced to expand the receptive field and scale down the frame size for compression. After that, we up-sample the number of frames with ratio U by

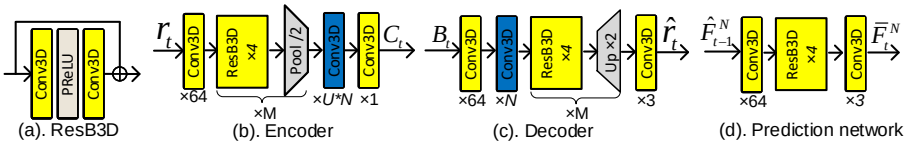


Fig. 2: Network architectures of sub-modules. The yellow modules are built with Channel-Conv3D and the blue are with Frame-Conv3D. The number after × is either the amount of convolution filters or the replica of blocks. (a). The 3D resnet block (ResB3D). (b). The encoder with a Frame-Conv3D to up-sample frames. (c). The decoder with a Frame-Conv3D to down-sample frames. (d). The prediction network using ResB3D.

Frame-Conv3D for latter multi-frame differential modulation. The difference between Channel-Conv3D and Frame-Conv3D is discussed in section 3.2. Finally, we down-sample the channel number to 1 with Channel-Conv3D for compression. Thus the encoded tensor C_t before quantizer can be denoted in Eq. 1, and it should have a shape $C_t \in R^{1\times(U*N)\times H/2^M\times W/2^M}$, where M is the number of max-pooling with 2 kernel size.

$$C_t = E_\phi(r_t) = E_\phi(F_t^N - \bar{F}_t^N) = E_\phi(F_t^N - P_\sigma(\hat{F}_{t-1}^N)) \tag{1}$$

Step 3: quantization with multi-frame differential modulation. A quantizer is necessary to generate the bit-stream $B_t \in \{-1, 1\}^{s(C_t)}$ from float tensor C_t. Trainable quantizer has been well studied in previous works for model quantization [9] and image compression [4, 21]. However, these works mainly focus on how to propagate the gradients with this non-differentiable operation, but fail to make full use of the signal and noise characteristic. Instead, we introduce a differential quantizer to alleviate the effect of quantization noise. The key here is to quantize the differential information between up-sampled frames. Details will be given in section 3.3. After this step, the output bit-stream B_t have the same shape of C_t, i.e. $B_t \in R^{1\times(U*N)\times H/2^M\times W/2^M}$, but each pixel of B_t is quantized into -1 or 1. To measure the number of bits for compressing, we use bits per pixel (BPP) to represent the required bits for each pixel in the current frames. It can be calculated that the maximum BPP is $\frac{U}{4^M}$ in our model.

Step 4: decoder. The decoder D_θ in Fig. 2(c) is nearly the inverse process of the encoder. First, we up-sample the channel from 1 to 64 with Channel-Conv3D and down-sample the frames from $U * N$ to the original N with Frame-Conv3D. Then we decode the side information using 3D resnet and scale up the features to the original size with bilinear interpolation. After that we down-sample the channel number to 3 to reconstruct the side information $\hat{r}_t$. Finally, the decoded N frames $\hat{F}_t^N$ at time step t can be described as Eq. 2.

$$\hat{F}_t^N = \bar{F}_t^N + \hat{r}_t = P_\sigma(\hat{F}_{t-1}^N) + \hat{r}_t = P_\sigma(\hat{F}_{t-1}^N) + D_\theta(B_t). \tag{2}$$

Output: the quantized bit-stream B_t and the reconstructed N frames $\hat{F}_t^N$.

3.2 Channel-Conv3D and Frame-Conv3D

Conv3D can learn the spatio-temporal features of multiple frames effectively. Traditional Conv3D sums up the partial results along the input channel dimension. The output can be denoted in top of Eq. 3, where $*$ is the convolution operation, and C_{out} is the number of 3D filters that determines the output channel. We denote this operation as Channel-Conv3D. The receptive field of one Channel-Conv3D layer is limited by the small kernel size which is usually 3. Multiple layers can extend the receptive field but introduce high computing cost. Therefore, we proposed the Frame-Conv3D that sums up the partial results along the input frame dimension. It is described in bottom of Eq. 3, where N_{out} is the number of output frames. It can be noticed that all features of input frames are fused to the output by summing operation regardless of the kernel size. We replace some traditional Channel-Conv3D with Frame-Conv3D in encoder and decoder to further fuse the features of multiple frames. Another potential benefit is that the number of output frames can be arbitrarily assigned, which is useful for the mult-frame differential modulation (discussed in section 3.3).

$$\begin{aligned} \text{Channel-Conv3D: } \mathrm{out}(\mathrm{C_{out_j}}) &= \mathrm{bias}(\mathrm{C_{out_j}}) + \sum_{\mathrm{i}=0}^{\mathrm{C_{in}}-1} \mathrm{weight}(\mathrm{C_{out_j}}) * \mathrm{input} \\ \text{Frame-Conv3D: } \mathrm{out}(\mathrm{N_{out_j}}) &= \mathrm{bias}(\mathrm{N_{out_j}}) + \sum_{\mathrm{i}=0}^{\mathrm{N_{in}}-1} \mathrm{weight}(\mathrm{N_{out_j}}) * \mathrm{input} \end{aligned} \tag{3}$$

3.3 Differential Quantizer Q

In order to generate a binary feature maps, previous works adopt a pixel-level quantizer. They quantize each pixel of the activation without considering the dependency between frames. Different from any previous methods, we propose a differential quantizer. It is based on the multi-frame differential modulation and can effectively eliminate the redundant information for binary compression. This section first gives the detailed implementation of the differential quantizer in DL framework, and then explains the effectiveness of this method in theory.

Implementation of differential quantizer in DL framework. The differential quantizer is composed by a differential modulator and a trainable quantizer Q_b. After we get the up-sampled frames $C_t = \{c_t[1], c_t[2], ..., c_t[U*N]\}, c_t \in R^{1\times H/2^M \times W/2^M}$ in section 3.1, we first apply differential modulation to each frame c_t and generated the modulated result y_t. Then a trainable quantizer is applied on y_t to generate the final binary output b_t.

1) **Trainable quantizer** Q_b. We employ the binary technique proposed in [21] to realize a trainable quantizer Q_b for DL framework. We first apply a tanh activation on the input to normalize it into the range of [-1, 1]. Then the binary function $B(x)$ for $x \in [-1, 1]$ is defined as $B(x) = x + \epsilon$, $B(x) \in \{-1, 1\}$, ϵ has $(1 + x)/2$ probability to be $1 - x$ and $(1 - x)/2$ probability to be $-1 - x$. The ϵ is the quantization noise. Therefore, the full function of the quantizer Q_b according to the modulated signal y_t is $Q_b(y_t) = B(\tanh(y_t))$. For the backward pass of gradients, the derivative of

the expectation is taken [21]. Since $\mathbb{E}[B(x)] = x$, the gradients pass through B is unchanged. Once the network are trained, $B(x)$ is replaced by the sign function to get a fixed representation for a particular input.

2) **Forward computing of the differential quantizer Q.** Based on the trainable quantizer Q_b, the forward computing flow of the differential quantizer Q is presented in Algorithm 1. The input C_t has been up-sampled with Frame-Conv3D and each c_t represents a frame in C_t. Q_b is the defined trainable quantizer, $Y_t = \{y_t[1], ..., y_t[n]\}$ are the modulated frames before quantizer and $B_t = \{b_t[1], ..., b_t[n]\}$ are the quantized outputs. Line 3 in Algorithm 1 is the definition of the differential modulation. It should be noticed that the modulation is not simply the difference between $c_t[i]$ and $c_t[i-1]$. The current $y_t[i]$ is modulated not only with $c_t[i]$, but also the $y_t[i-1]$ and $b_t[i-1]$ of last frame. We theoretically identify that the modulator can effectively deal with the quantization noise, which is explained in next part of this section. Since the computing flow only includes a trainable quantizer and operations of addition and subtraction, it is compatible with the DL framework and can be optimized by back propagation algorithm.

Algorithm 1 The forward computing flow of differential quantizer Q

Input: Up-sampled frames $C_t = \{c_t[1], c_t[2], ..., c_t[n]\}$
Output: Quantized output $B_t = \{b_t[1], b_t[2], ..., b_t[n]\}$
1: **Initialization** $y_t[0] \Leftarrow 0$, $b_t[0] \Leftarrow 0$
2: **for** i from 1 to n **do**
3: $\quad y_t[i] \Leftarrow y_t[i-1] + c_t[i] - b_t[i-1]$ (Differential Modulator)
4: $\quad b_t[i] \Leftarrow Q_b(y_t[i])$ (Trainable Quantizer)
5: **end for**

3) **Back propagation of the differential quantizer Q.** To see how the weights update during back propagation with differential modulation, we calculate the gradients of Algorithm 1. In traditional quantizer, only Q_b has been used, i.e., $b'_t = B(\tanh(c_t))$, where $B(x)$ is the binarization function defined above and has derivative 1. The gradients propagated from c_t to b'_t is $\partial b'_t / \partial c_t = 1 - \tanh^2(c_t)$, which is pixel independent. For the differential quantizer Q, we can rewrite the updating function of line 3 and 4 in Algorithm 1 as

$$b_t[m] = Q_b(y_t[m]) = Q(c_t[m], c_t[m-1], ..., c_t[1]) \tag{4}$$

It can be noticed that $b_t[m]$ is updated not only from $c_t[m]$, but also all previous frames. So the gradients of Q should also propagate from latter frames to previous. The gradients from $b_t[m]$ to $c_t[m-k]$ can be calculated as Eq. 5 (detailed proof is given in the supplementary material).

$$\frac{\partial b_t[m]}{\partial c_t[m-k]} = \begin{cases} 1 - \tanh^2(y_t[m]), & k = 0 \\ (1 - \tanh^2(y_t[m])) \times \prod_{i=1}^{k} \tanh^2(y_t[m-i]), & 1 \le k \le m-1. \end{cases} \tag{5}$$

The first gradient at $k = 0$ is consistent with tradition methods that updating from $b_t[m]$ to $c_t[m]$. It's interesting to analyze the second gradients from $b_t[m]$ to $c_t[m-k]$. According to this equation, the $b_t[m]$ propagates gradients to all previous frames $c_t[m-k]$ with weight $w_k = \prod_{i=1}^{k} \tanh^2(y[m-i])$. This indicates that the network not only optimizes in pixel level, but also makes use of information in previous frames to remove the redundancy between frames. Therefore, this method is more likely to encode a binary distribution with less loss of information for compressing. Since $|\tanh^2(\mathrm{x})| < 1$, an important property of w_k is that $w_0 = 1 > w_1 > w_2 > ... > w_{m-1}$. This distribution of w is intuitively correct since two frames with long distance (larger k) should be less relevant and have smaller weight. This analysis also indicates the importance of tanh activation for network training.

Explanation for the effectiveness of differential quantizer. Based on the computing flow, we can theoretically analyze the effectiveness of the proposed differential quantizer. Notice that the trainable quantizer Q_b in line 4 of Algorithm 1 can be modeled by Eq. 6, where $B(x)$ is the defined binary function, $y_t[i]$ is the modulated frames before quantization, and $e[i]$ is the additive noise. To simplify the formula, we adopt a linear approximation to get the last equality with an equivalent noise $e'[i]$. We identify that in a well-trained network, $y_t[i]$ often have values in $[-1, 1]$. Since $\tanh(y_t[i]) - y_t[i]$ is close to zero when $y_t[i]$ is in $[-1, 1]$, this approximation is acceptable.

$$\begin{aligned} Q_b(y_t[i]) &= B(\tanh(y_t[i])) = \tanh(y_t[i]) + e[i] \\ &= y_t[i] + (\tanh(y_t[i]) - y_t[i] + e[i]) = y_t[i] + e'[i] \end{aligned} \tag{6}$$

Combining Eq. 6 with the updating function of the differential modulator in Algorithm 1 (line 3), we can subsequently infer the following equation,

$$b_t[i] = c_t[i] + e'[i] - e'[i-1] \tag{7}$$

Now it's obviously to see how the differential modulation deal with the quantization noise from Eq. 7. It differential modulates the quantization noise between $e'[i]$ and $e'[i-1]$ to generate the quantized output b_t. So the modulator can effectively alleviate the effect of quantization. More specifically, if we adopt a U-tap moving average filter on b_t to recover r^{th} frame $\hat{c}_t[r]$, i.e.,

$$\hat{c}_t[r] = \frac{1}{U} \sum_{i=r+1}^{r+U} b_t[i] = \frac{1}{U} \sum_{i=r+1}^{r+U} c_t[i] + \frac{1}{U}(e'[r+U] - e'[r]) \tag{8}$$

The first item of Eq. 8 is the average of input c_t, and the second is accumulated noise. The noise comes to zero with large U. U is actually corresponding to the up-sample ratio by the Frame-Conv3D mentioned in section 3.1. In implementation, we replace the average filter with trainable convolution to improve the performance. The analysis shows that combing Frame-Conv3D and differential quantizer together can effectively reduce the quantization noise. In fact, this up-sampling and modulation principle is similar to the theory of 1st order $\Delta\Sigma$ modulation [18] in design of analog-to-digital converter (ADC) circuits. Although signal in $\Delta\Sigma$ modulation is usually one-dimensional while the frame is

3-dimensional in video, the method of frequency-domain analysis in $\Delta\Sigma$ can be adopted to visually understand the benefit of differential modulation. Fig. 3(a) shows the computing flow of $\Delta\Sigma$ modulation at current time step n in discrete time domain, which is similar to our proposed differential quantizer. In frequency domain, we can compute the signal transfer function (STF) and noise transfer function (NTF). The power spectral density (PSD) of STF and NTF is plotted in Fig. 3(b). The NTF of the modulator is a high-pass filter function which shapes noise e from low frequency band to high. Owing to this noise shaping characteristic, the in-band noise after quantizaion is suppressed and the signal-to-noise ratio (SNR) is improved greatly by adopting $\Delta\Sigma$ modulation, which is shown in Fig. 3(c). Although the same theory can be adopted to analyze the noise effect, we should mention that the frameworks of these two algorithms are totally different. $\Delta\Sigma$ modulation is a forward flow while the proposed multi-frame differential modulation needs to back propagate gradients. The techniques in $\Delta\Sigma$ modulation cannot be simply copied into DL frameworks. For example, higher order $\Delta\Sigma$ modulation can achieve better performance of noise shaping in ADC. However, we identify that using higher order differential modulation cannot achieve improvement, which is discussed in section 4.

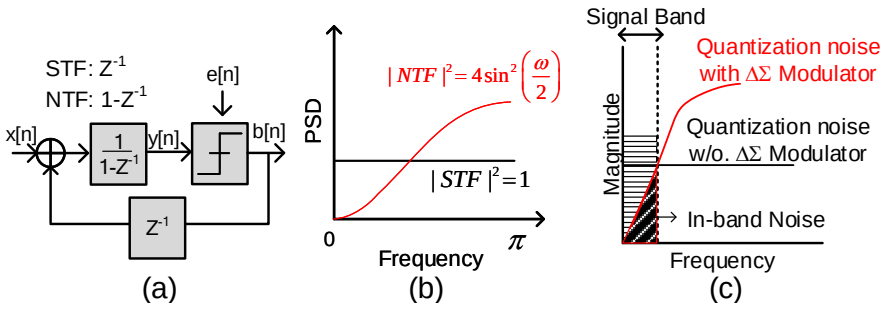


Fig. 3: Theory of the 1st order $\Delta\Sigma$ modulation. (a). Computing flow in Z domain. (b). The power spectral density (PSD) of NTF (red) and STF (black). (c). Noise shaping of $\Delta\Sigma$.

3.4 Single Model Supporting Multiple Bit Rate

The rate-distortion optimization is significant for video compression. Inspired from the forward and backward flows of differential modulation, we propose a dropout scheme for bit rate optimization. Noticing that the mth quantized frame $b[m]$ receives information from all previous frames $\{c_t[1], ..., c_t[m]\}$, once some later frames $\{c_t[k], c_t[k+1], ..., c_t[m]\}$ are dropped, the former $\{c_t[1], ..., c_t[k]\}$ can still help to update the $b_t[m]$. As a result, we can directly dropout some latter up-sampled frames to decrease bit cost for rate optimization. From experimental results in section 4.2 we find that the proposed differential quantizer can still recover high-quality video frames while traditional methods are failed in case of

dropping frame bits. Moreover, this scheme is similar to the dropout training and is compatible with DL framework. Involving this scheme in training can further improve the network performance at different levels of bit rate.

3.5 Training Strategy

To reduce the distortion between original frames F_t^N and reconstructed $\hat{F}_t^N$, we adopt the following loss function:

$$\mathcal{L} = d(F_t^N, \hat{F}_t^N) + \lambda d(F_t^N, \bar{F}_t^N) \tag{9}$$

where $d(F_t^N, \hat{F}_t^N)$ is the reconstructed distortion and $d(F_t^N, \bar{F}_t^N)$ is to control the distortion of prediction network. We find that the second loss can help to achieve some improvement in PSNR. In implementation, we use l_1 loss to measure the distortion and set λ as 0.05. Since previous reconstructed frames $\hat{F}_{t-1}^N$ are required for training,we adopt the same online updating strategy proposed in [14]. The mini-batch size is set as 8 and the training patch size of input frames is 128×128. We use the Adam optimizer [11] by setting the initial learning rate (lr) as 2e-4, β_1 as 0.9, and β_2 as 0.99. The lr is divided by 2 at every 50k iterations.

4 Experiments

We train the models with ZuriVID dataset, which is adopted in AIM 2019 challenge [1] for video extreme super-resolution. The dataset has 50 1080p videos with a total number of 69604 frames. The UVG dataset [2] and the HEVC Standard Test Sequences (Class B, Class C, Class D, and Class E) [19] are used to evaluate our model.

4.1 Network Parameters

To search for suitable parameters of coding frame number N and the down-scale times M with max-pooling, control experiments are implemented. For fairly comparison, the BPPs of all models are set the same as 0.25. A subset of UVG dataset including 224 frames of 7 videos is used for validation during training. We analyze four cases and the training results are shown in Fig. 4(a). We find that fewer frame number of 4 can converge faster than 8, and both achieve a similar performance at last. Down-scaling the feature to $1/4 \times 1/4$ with $M = 2$ has a better performance than down-scaling to $1/8 \times 1/8$ with $M = 3$. We also identify that tanh activation is important for back propagation. Based on the experiments, we set N as 4 and M as 2 in our final model.

The paper is focused on the proposed differential quantizer in 1st order. To confirm whether higher order modulation takes effect, we compare 1st and 2nd order differential modulation. The 2nd order modulator utilizes former two frames to modulate the current, and the detailed implementation is provided in the supplementary material. The results of evaluated PSNR and MS-SSIM are

presented in Fig. 4 (b). It shows that introducing higher order modulator takes little effects, and may even cause a decreasing in metrics. Theoretically, although 2nd order modulator uses extra information at $n-1$ and $n-2$ to update the current $c_t[n]$ while 1st order only use $n-1$, the updating paths of gradients are similar, i.e., propagating from $b_t[n]$ to $c_t[n]$ and all previous $c_t[n-1], ..., c_t[1]$. So they should get similar results ultimately with the back propagation algorithm. However, high order modulation introduces higher computing costs and makes the rule of updating gradients extremely complex. From this perspective, 1st order $\Delta\Sigma$ quantizer is superior.

Besides, we also remove the prediction network or the second prediction loss $d(F_t^N, \bar{F}_t^N)$ in Eq. 9 for further comparison. We notice that either removing the prediction network or loss can substantially decreasing the performance, as shown in Fig. 4 (b). This suggests that motion estimation and compensation from previous decoded frames to currents is important for video compression. The current prediction network is a multi-layer 3D CNN. To improve the performance, more complex motion estimation model can be considered, such as models with optical flow and block motion vectors.

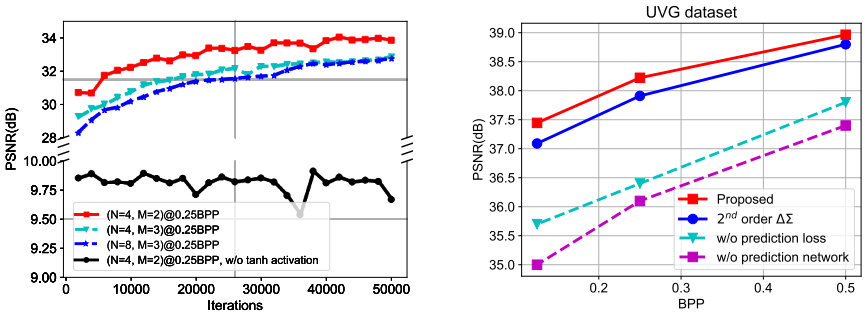


Fig. 4: (a). Evaluated PSNR during training with different parameters: frame number N and down-scale times M. (b). Comparison between models of the 1st order differential quantizer (proposed), the 2nd order differential quantizer, and removing the prediction loss.

4.2 Advantages of Differential Quantizer with Visual Results

To illustrate the advantages of differential quantizer in visual, we train 3 models for ablation study, i.e., up-sampling and down-sampling in encoder and decoder by:

(A). only using Channel-Conv3D with traditional quantizer Q_b;
(B). using Frame-Conv3D with traditional quantizer Q_b;
(C). using Frame-Conv3D with differential quantizer Q.

For a better visual comparison, we include more frames as input and set N as 8 to show the efficient frame fusion of differential modulation. In order to

better visualize the middle quantized results in case of dropping frames, the prediction network is removed when comparing between model A/B/C. All hyper-parameters and model blocks are set with no difference for these 3 models except for the settings of quantizer. Model A replaces all the Frame-Conv3D with $\times 64$ Channel-Conv3D and change the last Conv3D to $\times U$ in Fig. 2(b) to guarantee the same BPP level with model B and C. We set the number of max-pooling M as 2 and the up-sampling ratio U as 16 and 4, which results in 1 and 0.25 maximum BPP, respectively. All models are trained for up to 50×10^3 iterations.

Fig. 5 shows the evaluated PSNR of the subset in training process. It can be observed that model B with Frame-Conv3D trains better than model A. Model C with differential quantizer can improve the metric significantly within 50k training iterations, especially at low BPP, e.g., 0.25. The curves show that the proposed method can effectively train a network for binary representation, which has been analyzed in section 3.3.

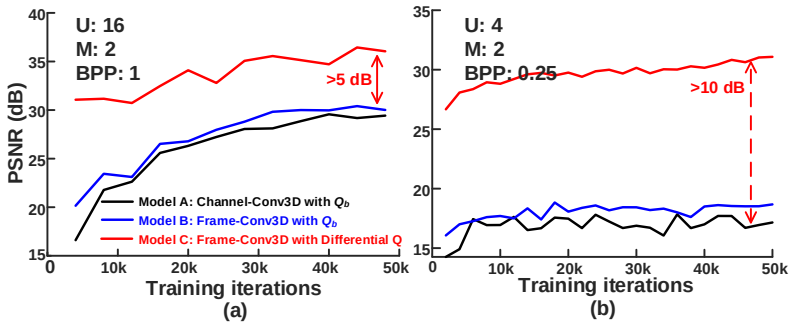
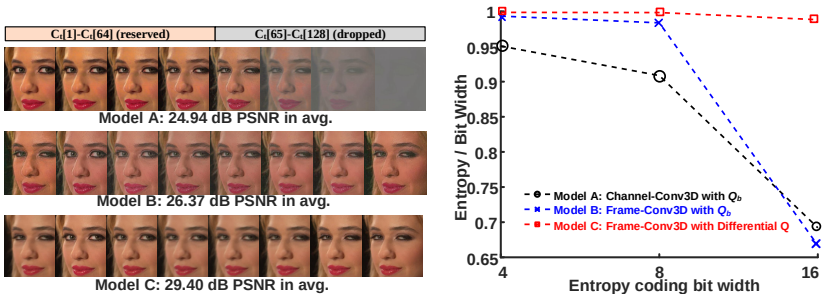


Fig. 5: Evaluated PSNR of model A,B, and C during the training phase. (a). One quarter down-scaled with 1 BPP. (b). One eighth down-scaled with 0.25 BPP.

To identify the effectiveness of differential quantizer in case of dropping bits, we evaluate the well-trained three models only with former half frames at case of $U = 16$. Since U is 16 and N is 8, there are 128 up-sampled frames in total. We reserve the former 64 frames and set the latter 64 frames to zero, which is shown in Fig. 6(a). Model A cannot fuse the frame information properly and the latter recovered frames will be noisy gradually. The average PSNR decreases up to 10 dB. Model B can reconstruct all frames because of the efficient feature fusion along the frame dimension with Frame-Conv3D. However, the reconstructed frames still lose much detailed information, resulting in a low video quality according to PSNR (around 8 dB loss). For model C with differential quantizer, all frames are decoded in good quality and the decreasing in PSNR is less than 5dB. It should be mentioned that all the above models are not particularly optimized in training phase for the case of dropping bits. It shows that differential modulation can make full use of all past frame information for recovering. Adopting dropout scheme in training phase can certainly further improve the performance.

The entropy of generated bit streams for each model is also calculated and compared in Fig. 6(b). When coding with different bit width, model C achieves

the highest entropy and there is nearly no room for compression, which means it successfully remove the redundant information between frames and no extra entropy coding is needed. However, margins of compression still exist for the cases without differential modulation. When the coding bit width is 16, model A and B can further achieve nearly 30% compressing gain by entropy coding.



(a) (b)

Fig. 6: Experimental comparison between Model A, B, and C. (a). Reconstructed frames of three models in case of dropping bits. (b). Entropy of bit-streams generated by three models.

4.3 Comparison to Previous Works

We train the final model in two steps. First, a model based on differential quantizer is obtained by 100k training iterations without dropout scheme. Then the pre-trained model is re-trained combined with dropout scheme for bit rate optimization. Although we can drop arbitrary number of frames, in implementation we only define 3 fixed patterns that drop {50%, 75%, 87.5%} frames of all to speed up training. As a result, the trained single model can encode videos into {0.5, 0.25, 0.125} BPPs respectively. We compare our method with H.264, H.265 and two DL-based methods of [14, 26] in terms of PSNR and MS-SSIM metrics. We follow the setting in [14] and use the FFmpeg tool with very fast mode to generate the compressed frames of H.264 and H.265.

We find that our model achieves high performance on UVG and HEVC Class D and Class E dataset, which is shown in Fig. 7. Compared with H.264 and H.265 codecs on the three dataset, the model shows superiority especially in low BPP region. It also outperforms the DL-based method of [26] and is competitive with [14] on UVG dataset. It should be noticed that our method jointly trains a single model to support different BPP levels, while the work by Lu *et al.* [14] needs to train multiple models for each BPP level.

However, our model shows some weaknesses on HEVC Class B and Class C dataset (results are provided in supplementary materials) because of the fast moving objects (BasketballDrill, BasketballDrive, RaceHorses, etc) in such videos. It can be improved by adopting optical flow or motion vectors in the prediction network. Despite this, we pay more attentions to the efficient training method for the quantizer and achieve high performance on vast majority of

videos. Other techniques for the motion estimation can be easily plugged into this framework to further improve the performance, which is our future work.

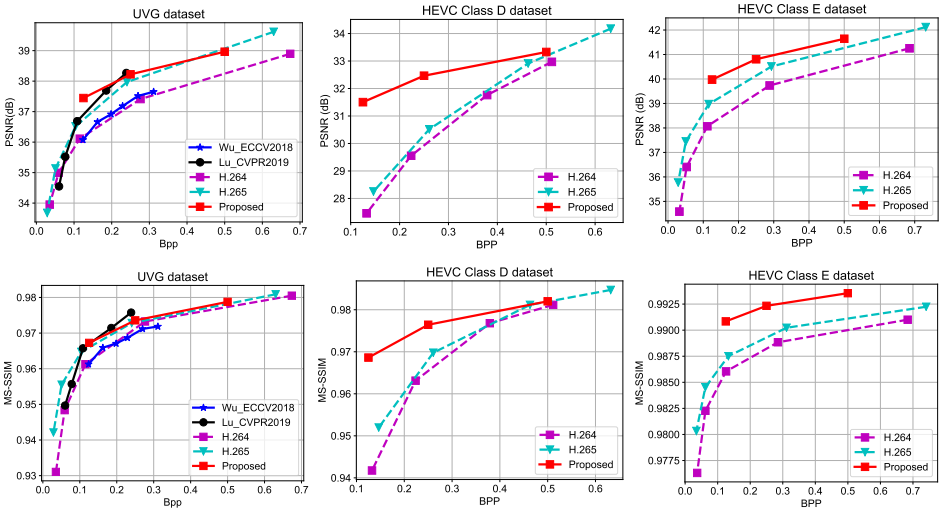


Fig. 7: Comparison between our model and H.264 [25], H.265 [19], the DL-based video method in [26] and [14]. The data for DL-based codecs are from paper report, and the H.264 and H.265 results are from FFmpeg tool.

5 Conclusions

This paper proposes an end-to-end model for deep video compression. We introduce the Frame-Conv3D and multi-frame differential modulation in codecs to reconstruct high-quality video. Different from traditional quantizer with pixel-level binarization, the proposed method of differential modulation can quantize the difference and propagate gradients between frames to remove the redundant information as much as possible. We identify the differential quantizer can alleviate the effect of quantization noise by theoretical analysis and experimental results. Inspired from the gradient propagating path of differential quantizer, we propose a dropout scheme to optimize bit rate within single model. Experimental results show that the proposed model outperforms standard H.264 and H.265 in low BPP region and is competitive with the recent deep codecs. Based on the proposed model, other techniques of optical flow or motion vector estimation can be easily plugged into this framework to further improve the performance.

Acknowledgment. This work is supported in part by National Key R&D Program (2019YFB2204204), NSFC Grant (No. 61934005,61674094), Beijing National Research Center for Information Science, Technology, and Beijing Innovation Center for Future Chip.

References

1. Advances in image manipulation workshop and challenges on image and video manipulation. http://www.vision.ee.ethz.ch/aim19/, accessed November 10, 2019
2. Ultra video group test sequences. http://ultravideo.cs.tut.fi, accessed November 7, 2019
3. Agustsson, E., Tschannen, M., Mentzer, F., Timofte, R., Van Gool, L.: Generative adversarial networks for extreme learned image compression. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 221–231 (Oct 2019). https://doi.org/10.1109/ICCV.2019.00031
4. Agustsson, E., Mentzer, F., Tschannen, M., Cavigelli, L., Timofte, R., Benini, L., Gool, L.V.: Soft-to-hard vector quantization for end-to-end learning compressible representations. In: Advances in Neural Information Processing Systems. pp. 1141–1151 (2017)
5. Ballé, J., Laparra, V., Simoncelli, E.P.: End-to-end optimized image compression. arXiv preprint arXiv:1611.01704 (2016)
6. Ballé, J., Minnen, D., Singh, S., Hwang, S.J., Johnston, N.: Variational image compression with a scale hyperprior (2018)
7. Caballero, J., Ledig, C., Aitken, A., Acosta, A., Shi, W.: Real-time video super-resolution with spatio-temporal networks and motion compensation (2016)
8. Cisco, V.: Cisco visual networking index: Forecast and trends, 2017–2022. White Paper **1** (2018)
9. Courbariaux, M., Hubara, I., Soudry, D., Elyaniv, R., Bengio, Y.: Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. arXiv: Learning (2016)
10. Diba, A., Pazandeh, A.M., Van Gool, L.: Efficient two-stream motion and appearance 3d cnns for video classification. arXiv preprint arXiv:1608.08851 (2016)
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv: Learning (2014)
12. Li, M., Zuo, W., Gu, S., Zhao, D., Zhang, D.: Learning convolutional networks for content-weighted image compression. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3214–3223 (June 2018). https://doi.org/10.1109/CVPR.2018.00339
13. Li, S., He, F., Du, B., Zhang, L., Xu, Y., Tao, D.: Fast spatio-temporal residual network for video super-resolution. arXiv preprint arXiv:1904.02870 (2019)
14. Lu, G., Ouyang, W., Xu, D., Zhang, X., Cai, C., Gao, Z.: Dvc: An end-to-end deep video compression framework. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10998–11007 (June 2019). https://doi.org/10.1109/CVPR.2019.01126
15. Minnen, D., Ballé, J., Toderici, G.: Joint autoregressive and hierarchical priors for learned image compression (2018)
16. Rippel, O., Bourdev, L.: Real-time adaptive image compression. arXiv: Machine Learning (2017)
17. Rippel, O., Nair, S., Lew, C., Branson, S., Anderson, A.G., Bourdev, L.: Learned video compression. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3454–3463 (2019)
18. Schreier, R., Pavan, S., Temes, G.C.: Understanding delta-sigma data converters —— the delta-sigma toolbox **10.1002/9781119258308**, 499–537
19. Sullivan, G.J., Ohm, J.R., Han, W.J., Wiegand, T.: Overview of the high efficiency video coding (hevc) standard. IEEE Transactions on Circuits & Systems for Video Technology **22**(12), 1649–1668

20. Theis, L., Shi, W., Cunningham, A., Huszár, F.: Lossy image compression with compressive autoencoders (2017)
21. Toderici, G., O'Malley, S.M., Hwang, S.J., Vincent, D., Minnen, D., Baluja, S., Covell, M., Sukthankar, R.: Variable rate image compression with recurrent neural networks. Computer Science (2015)
22. Toderici, G., Vincent, D., Johnston, N., Jin Hwang, S., Minnen, D., Shor, J., Covell, M.: Full resolution image compression with recurrent neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5306–5314 (2017)
23. Tong, C., Liu, H., Qiu, S., Tao, Y., Zhan, M.: Deepcoder: A deep neural network based video compression. In: 2017 IEEE Visual Communications and Image Processing (VCIP) (2017)
24. Wang, Z., Simoncelli, E.P., Bovik, A.C.: Multiscale structural similarity for image quality assessment. In: Signals, Systems and Computers, 2003. Conference Record of the Thirty-Seventh Asilomar Conference on (2003)
25. Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A.: Overview of the h. 264/avc video coding standard. IEEE Transactions on circuits and systems for video technology **13**(7), 560–576 (2003)
26. Wu, C.Y., Singhal, N., Krahenbuhl, P.: Video compression through image interpolation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 416–431 (2018)