Hierarchical B-frame Video Coding Using Two-Layer CANF without Motion Coding

David Alexandre[1], Hsueh-Ming Hang[2], Wen-Hsiao Peng[3]
[1]Electrical Engineering and Computer Science International Graduate Program
[2]Institute of Electronics
[3]Department of Computer Science
National Yang Ming Chiao Tung University
Hsinchu, Taiwan
{davidalexandre.eed05g,hmhang}@nctu.edu.tw, wpeng@g2.nctu.edu.tw

Abstract

Typical video compression systems consist of two main modules: motion coding and residual coding. This general architecture is adopted by classical coding schemes (such as international standards H.265 and H.266) and deep learning-based coding schemes. We propose a novel B-frame coding architecture based on two-layer Conditional Augmented Normalization Flows (CANF). It has the striking feature of not transmitting any motion information. Our proposed idea of video compression without motion coding offers a new direction for learned video coding. Our base layer is a low-resolution image compressor that replaces the full-resolution motion compressor. The low-resolution coded image is merged with the warped high-resolution images to generate a high-quality image as a conditioning signal for the enhancement-layer image coding in full resolution. One advantage of this architecture is significantly reduced computational complexity due to eliminating the motion information compressor. In addition, we adopt a skip-mode coding technique to reduce the transmitted latent samples. The rate-distortion performance of our scheme is slightly lower than that of the state-of-the-art learned B-frame coding scheme, B-CANF, but outperforms other learned B-frame coding schemes. However, compared to B-CANF, our scheme saves 45% of multiply–accumulate operations (MACs) for encoding and 27% of MACs for decoding. The code is available at https://nycu-clab.github.io.

1. Introduction

Digital video compression has been studied for over 50 years. It is a challenging research topic to exploit both spatial and temporal redundancies inside the video data. The concept of using motion compensation to reduce temporal correlation for video coding first appeared in 1969 [30]. Since then, motion estimation and coding have become indispensable components in a video coding system. Two critical components in a mainstream video codec are motion coding (including motion estimation and compensation) and residual image coding. Motion coding is used to reduce *temporal* redundancy, and residual coding is used to reduce *spatial* redundancy. This structure is thus often called *hybrid coding*. The influential and widespread international video standards in the past three decades, MPEG-2, AVC/H.264, HEVC/H.265, and VVC/H.266 all adopt this basic hybrid coding structure, although the fine details vary in different versions of standards. These standards specify three types of coding frames inside a Group of Pictures (GOP): I-frame (intra-coded), P-frame (predictive), and B-frame (bidirectional predictive). The P-frame coding process uses the previously coded frame to predict the target frame, and the B-frame coding uses two reference frames (often previous and future frames) to predict the target frame. In this paper, we focus on learning-based B-frame video coding.

In the past few years, deep-learning techniques have been used in video compression. Up to now, most learned codecs adopt the hybrid coding structure of the classical coding systems; that is, it contains two major components: motion coding and residual image coding. It is generally believed that accurate motion compensation is a very effective way to reduce the temporal redundancy in the video. Only the remaining unpredictable ('new') pixels are coded using image coding techniques. Describing accurately the motion field around arbitrary shape objects often needs a large number of bits. For example, the HEVC standard defines a variety of block partitions to specify regions sharing the same motion vectors [31].

Thanks to the advancement of neural networks, more accurate video predictors without transmitting bits are now available. Then, we need only to send the unpredictable pixels. Often the locations of unpredictable pixels are sparse. It costs many bits to send the precise location information. Hence, we develop a bootstrap strategy. Instead of transmitting motion or location information or both, we send the unpredictable pixel information in two layers. The *base layer* sends the downsampled unpredictable information (containing locations and pixel values) to the decoder. This piece of information serves two purposes. It provides a rough, downsampled image of unpredictable pixels and contains information indicating which pixels are unpredictable. With a well-designed neural network, we generate a weighting map that merges predictable and unpredictable pixels to construct a good-quality target frame. Then, at the *enhancement layer*, we send additional information (bits) to improve the quality of the final coded image.

Motivated by the above observations, we propose a learned video compression scheme without a motion coding module. It contains two image coding layers: the base and enhancement layers. The base layer consists of a video frame interpolator, a downsampling network, a neural network-based image compressor, and a super-resolution network (SR-Net). We adopt the efficient Conditional Augmented Normalization Flows (CANF) [15] for the image compressors at the base and enhancement layers. The frame interpolator produces the conditioning image for the base-layer CANF. The SR-Net upsamples the decoded base-layer image to recover a full-resolution image. The enhancement layer consists of a multi-frame merging network, skip-mask generator, skip-mode coding module and CANF compressor. The multi-frame merging network combines all the image information available at both the encoder and the decoder to form a *merged* image. The merged image serves as the conditioning signal for the enhancement-layer CANF. To this end, we design a merging map (weights) generator, a neural network accepting inputs from the upsampled base-layer image, and two motion-warped reference frames. To improve the coding efficiency of the enhancement-layer compressor, we design a skip-mode coding technique. A neural network generates a binary skip mask SM_t according to the predicted motion information, the base-layer merged output, and the enhancement-layer hyperprior output. The skip mask specifies the locations of significant and insignificant latent samples. The insignificant samples are skipped from coding; at the decoder, they are replaced by the corresponding mean values predicted by the enhancement-layer hyperprior module. The detailed skip-mode coding operation is described in the supplementary document.

Our contributions are summarized as follows.

- We propose a two-layer B-frame coding framework that skips motion information from coding.
- We introduce a multi-frame merging network to combine the base-layer and enhancement-layer frames in constructing a high-quality predictor for the enhancement-layer CANF compressor.

We implement the above ideas in an end-to-end learned B-frame video compression system. Because the input image to the base-layer compressor has a much smaller dimension, our system has much lower computational complexity (about 45% lower in terms of encoding MACs) than B-CANF [10], a typical hybrid coding system with similar coding components.

2. Related Works

2.1. Deep Video Compression

Most existing deep video compression schemes adopt the hybrid coding structure with motion and residual coding, and focus on P-frame coding. For example, an early work by Lu et al. [26, 27] presents an efficient deep video coding scheme that replaces nearly all the key components in the classical coding architecture by deep neural networks. Recent deep video compression papers often use similar hybrid coding structures and focus on improving various components, e.g. motion coding [4, 16, 24], residual coding [13], feature-space coding [18], content-adaptive coding [25], coding mode prediction [17], and contextual coding [15, 22, 23]. One notable trend is the use of conditional/contextual coding to replace traditional residual coding. For example, the contextual coding papers [15, 22, 23] adopt this concept to achieve high coding performance.

There are only a few attempts at eliminating motion coding (i.e. not transmitting motion information) in a video codec [11], [36, 37], [9]. Both Cheng et al. [11] and Zou et al. [36, 37] adopt the hierarchical B-frame GOP structure, where Cheng et al. [11] encode frame differences. They adjust the temporal distance in calculating frame differences based on the motion characteristics. Zou et al. [36, 37] compute the pyramid features of the reference and target frames and derive motion information from the transmitted features at the decoder. On the other hand, Chen et al. [9] focus on P-frame coding and transmit the displaced frame differences instead of sending motion information. This concept of video compression without motion coding received little attention. The reason may be because of its inferior coding performance, although employing only one compressor significantly reduces the complexity. As discussed earlier, we observe that a low-rate base layer is needed to efficiently convey unpredictable pixels to improve compression performance.

Up to now, only a handful of deep video compression schemes address B-frame coding. In addition to [11, 36, 37], Wu et al. [32] introduce an early deep video compression

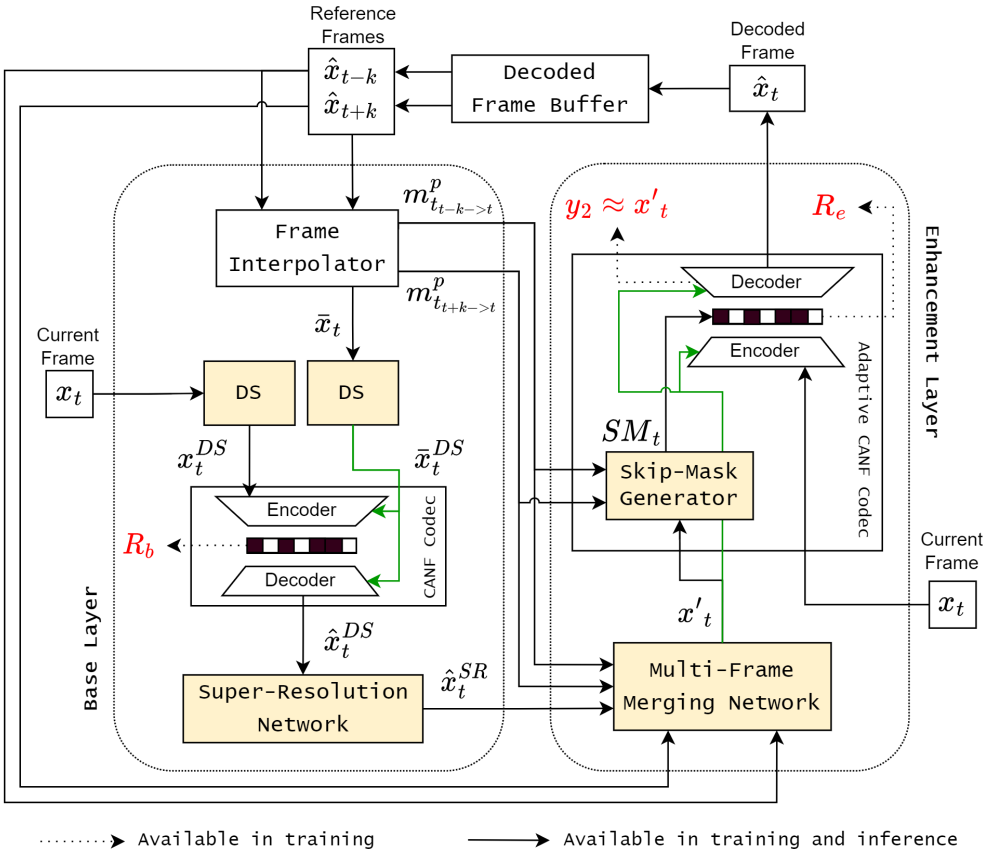


Figure 1. The proposed two-layer conditional B-frame coding system without motion coding. It includes a low-resolution CANF compressor and a full-resolution adaptive CANF compressor. The input frame x_t is encoded based on its reference frames $\hat{x}_{t-k}$, $\hat{x}_{t+k}$, with the decoded frame $\hat{x}_t$ representing a lossy reconstruction of x_t. The yellow blocks denote our proposed components. The green solid lines represent the conditioning signals for the CANF compressors. The red symbols are available only in the training phase.

system that encodes B-frames hierarchically using a simple image interpolation method. Often, the motion information for the two reference frames are coded and transmitted [12], [34]. Pourreza et al. [29] extend the P-frame coding method to encode B-frames using only one motion field. Yilmaz et al [35] propose learned hierarchical bi-directional video compression (LHBDC) that employs a temporal motion vector predictor to reduce the motion bitrate. It produces impressive coding performance when compared to the prior learned P-frame and B-frame codecs. This scheme was refined and extended to flexible rate compression by Çetin et al. [8].

2.2. CANF Compressors

Ho et al. [15] propose Conditional Augmented Normalizing Flows (CANF) [15] by combining the concept of conditional coding with an efficient deep image compression architecture, Augmented Normalizing Flows (ANF) [14]. In theory, conditional coding is more efficient than the residual coding that has been used in typical hybrid coding systems [21], [15]. Therefore, several conditional coding structures [22], [23], [15] are proposed, showing promising compression performance. CANF can replace the usual VAE compressors in the hybrid coding structure and produce the state-of-the-art performance in P-frame coding [15]. Recently, Chen et al. [10] apply CANF to B-frame coding. They still use the hybrid coding structure and show the state-of-the-art performance with an additional frame-adaptive coding technique. We thus also use CANF as the image compressor in our system, but we do not use the hybrid coding structure. Our adaptive CANF differs from the basic CANF (shown in the supplementary document) in that it incorporates a skip-mask generator and a skip-coding mechanism.

3. Proposed Method

3.1. System Overview

Figure 1 shows our two-layer conditional coding scheme for B-frame video compression. The basic building block of our system is CANF [15], and our intra-coding is an ANF image compressor from [14]. Our framework has two coding layers: a low-resolution CANF compressor (the base layer) and a full-resolution CANF compressor (the enhance-

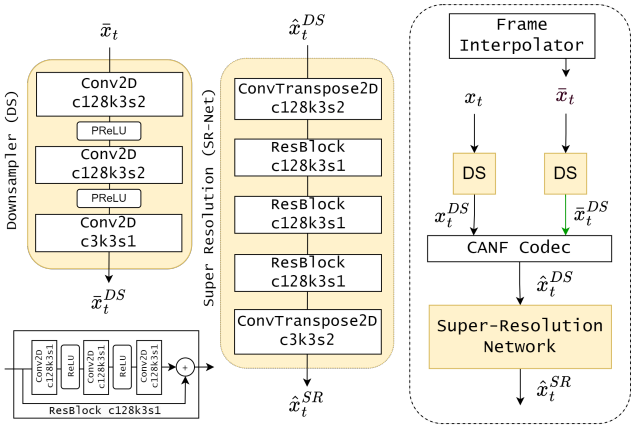


Figure 2. Illustration of the base-layer components. The yellow blocks indicate our proposed modules, i.e. the downsampler (DS) and super-resolution network (SR-Net). The high-resolution input image $\bar{x}_t$ to the downsampler is produced by the RIFE frame interpolator. It is downsampled by a factor of 4, with the resulting signal $\bar{x}_t^{DS}$ serving as the conditioning signal for the low-resolution CANF compressor, which encodes the downsampled version x_t^{DS} of the target frame x_t. The compressor output $\hat{x}_t^{DS}$ is upsampled by SR-Net as $\hat{x}_t^{SR}$.

ment layer).

3.2. Base Layer

The base layer comprises a frame interpolator, downsampler (DS), super-resolution network (SR-Net), and CANF compressor. We adopt an off-the-shelf high-performance video interpolator, RIFE [19], as our frame interpolator.

As shown in Figure 2, the downsampling network (DS) downsamples the pixel-domain interpolated frame $\bar{x}_t \in R^{3\times H\times W}$ to $\bar{x}_t^{DS} \in R^{3\times H/4\times W/4}$, where W, H are the width and height of the target frame $x_t \in R^{3\times H\times W}$, respectively. The same DS is also applied to x_t to produce $x_t^{DS} \in R^{3\times H/4\times W/4}$. The downsampled interpolated frame $\bar{x}_t^{DS}$ then serves as the conditioning signal for the CANF compressor to compress the downsampled target frame x_t^{DS}. After the compression step, we recover the resolution of the coded downsampled target frame $\hat{x}_t^{DS} \in R^{3\times H/4\times W/4}$ to its original resolution by a super-resolution network (SR-net).

Downsampling Network (DS). The DS network is composed of convolutional layers and residual blocks (Figure 2). Specifically, we use two convolutional layers with stride 2 to achieve a downsampling factor of $m = 4$.

Base-Layer CANF Compressor. The base-layer CANF compressor encodes the downsampled target frame x_t^{DS} by taking the downsampled version $\bar{x}_t^{DS}$ of the interpolated frame $\bar{x}_t$ as a conditioning signal. Note that the base-layer CANF compressor is to be distinguished from another CANF compressor in the enhancement layer.

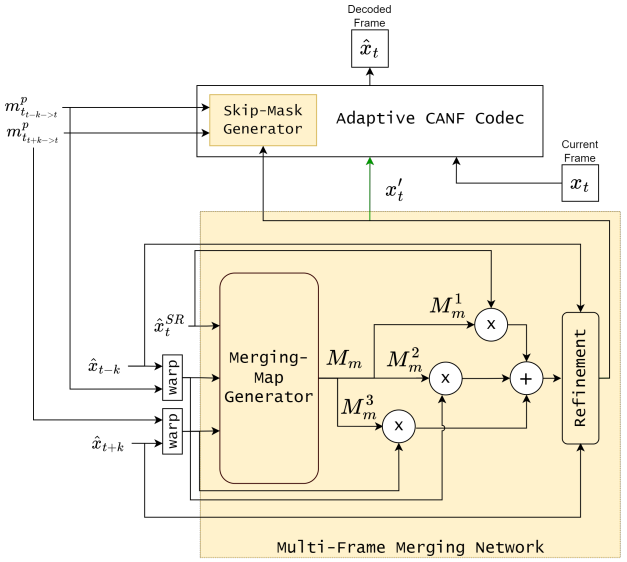


Figure 3. Illustration of the enhancement-layer components. The multi-frame merging network inside the enhancement layer is designed to combine $\hat{x}_t^{SR}$, $warp(\hat{x}_{t-k}, m^p_{t-k\to t})$, and $warp(\hat{x}_{t+k}, m^p_{t+k\to t})$ to produce a refined merged output x'_t according to a merging map generated by the merging-map generator.

Super-Resolution Network (SR-Net). The SR-Net is to interpolate the low-resolution coded target frame $\hat{x}_t^{DS}$ to its original resolution $\hat{x}_t^{SR} \in R^{3\times H\times W}$. We perform upsampling using transpose convolutions with stride 2. The network architecture is detailed in Figure 2.

3.3. Enhancement Layer

To obtain a high-quality output at the end of this stage, we introduce a multi-frame merging network. The network takes three inputs: the SR-Net output $\hat{x}_t^{SR}$ and the two warped (motion-compensated) reference frames $warp(\hat{x}_{t-k}, m^p_{t-k\to t})$, $warp(\hat{x}_{t+k}, m^p_{t+k\to t})$. It produces a floating-point weighting map $M_m \in R^{3\times H\times W}$ with three normalized values for each sample, which are the weightings used to combine the three input frames. The weighted sum of these three input frames is further refined using a refinement module (Refine-Net) to generate the final output image $x'_t \in R^{3\times H\times W}$. The architectures of the merging-map generator and Refine-Net are provided in the supplementary document.

We then use the second CANF compressor operating in the original image resolution to produce a high-quality coded frame. The merged output x'_t from the base layer serves as the conditioning signal to compress the target frame x_t. To reduce the bit consumption in arithmetic coding, we propose a skip-mode coding mechanism. The skip-mask generator network produces a binary skip-mask $SM_t \in \{0,1\}^{128\times H/16\times W/16}$ that determines which latent samples are transmitted in the arithmetic coding pro-

cess. We modify the CANF compressor from [14] to work with SM_t. The same skip mask is used at the decoder to identify the locations of non-skipped samples decoded from the transmitted bitstream. The reconstructed frame $\hat{x}_t \in R^{3\times H\times W}$ is stored in the decoded frame buffer and is used in the next coding cycle.

Multi-Frame Merging Network (MFMN). Inspired by [35], we develop a multi-frame merging network, which produces the weighting maps used to combine the output $\hat{x}_t^{SR}$ of SR-Net and the two warped reference frames, $warp(\hat{x}_{t-k}, m^p_{t_{t-k\to t}})$ and $warp(\hat{x}_{t+k}, m^p_{t_{t+k\to t}})$. Therefore, the output channel number is three and the softmax operator is used for scaling. Figure 4 illustrates the operations of our multi-frame merging network. In this example, the upper video is Jockey, and the lower one is HoneyBee. The three weighting maps $M^i_m \in R^{1\times H\times W}, i = 1, 2, 3$, the weighted output $M^1_m * \hat{x}_t^{SR}$ of the coded base layer $\hat{x}_t^{SR}$, and the final combined output x'_t are shown for one typical frame in these two sequences. The HoneyBee video is a slow-motion sequence; only a tiny honeybee has fast motion. Therefore, most of the background can be predicted well from the two reference frames. In contrast, both the object and background are moving in Jockey, and thus it is important to extract the locations of unpredictable pixels and their values from the decoded low-resolution image.

Skip-mask Generation. Our skip-mode coding mechanism has two main components: the (1) skip-mask generation and (2) skip-mode coding inside the arithmetic coder. The performance of the skip-mode coding largely relies on precise skip masks. Often the moving object boundaries and texture edges cannot be precisely predicted or upsampled from the low-resolution image. Hence, motion information provides clues to skip samples. Also, the decoded low-resolution image can provide object boundary and texture edge clues. Therefore, as shown in Figure 5, the first stage of our skip-mask generator takes inputs from the forward and backward motion fields, $m^p_{t-k\to t} \in R^{2\times H\times W}$, $m^p_{t+k\to t} \in R^{2\times H\times W}$, and the merged image, x'_t. We adopt the implementation of the skip-mask generation and skip-mode coding from [5].

Furthermore, the skipped (not transmitted) samples are replaced by the mean values μ from the hyperprior module at the decoder. This operation is performed also at the encoder to reconstruct the decoded image. The mean μ and variance σ produced by the hyperprior module also provide clues for skipping samples. Thus, the second stage of our skip-mask generator takes inputs from the hyperprior outputs, as shown in Figure 5. Finally, a rounding operation is applied to generate the binary mask. We use the straight-through gradient strategy in training to solve the zero gradient problem caused by the round operator for mask binarization. Value 0 in the skip mask means skip mode, and value 1 means non-skip mode. We show a few masks in Figure 6. Generally, more samples are skipped at lower bitrates.

Adaptive CANF Compressor. Because our enhancement-layer CANF includes the skip-mode coding process described above, it is called Adaptive CANF. The details of our Adaptive CANF is described in the supplementary document.

Frame-type Adaptive Coding. For better rate-distortion performance, *reference* B-frames should be coded with higher quality (at the cost of higher bitrates) than *non-reference* B-frames. To this end, we implement the frame-type adaptive (FA) coding proposed in [10]. Conceptually, the reference and non-reference B-frames are coded with two somewhat different models. This is achieved by applying a channel-wise affine transformation to the output features of every convolutional layer in our CANF compressors.

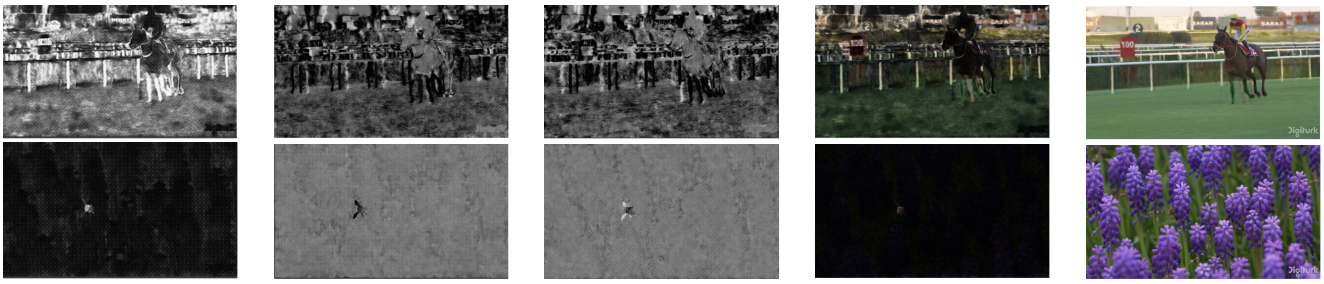
3.4. Training Procedure

Our model is trained by using a multi-step training procedure. The hyper-parameters are chosen empirically. We use the ADAM [20] optimizer with an initial learning rate of 1e-4. The batch size is set to 8. We train our model in four phases. Each phase has its own set of hyper-parameters and training loss functions. Some modules may be frozen during training; thus, only the other modules are trained in that step. Our training procedure is as follows.

1. We first train the frame interpolator (RIFE [19]) with the initial model from [19]. The loss function in this phase is $L = D(\bar{x}_t, x_t)$; that is, the output $\bar{x}_t$ of RIFE is trained to approximate the target frame x_t.

2. We train all the modules in the base layer in a few steps. The RIFE module is frozen at the beginning of this phase. First, we train only the downsampler and SR-Net (without the CANF compressor) using the loss function $L = D(\hat{x}_t^{SR}, x_t)$, where $\hat{x}_t^{SR}$ is the SR-Net output. When the first step reaches convergence, we include the CANF compressor between the downsampler and SR-Net in the second training step and the loss function is $L = D(\hat{x}_t^{DS}, x_t^{DS}) + R_b$, where R_b is the estimated coding bitrate of CANF in the base layer. Then, we train RIFE together with the downsampler, CANF, and SR-Net with $L = D(\hat{x}_t^{SR}, x_t) + R_b$ to update the entire base layer.

3. After the base layer produces a target frame with reasonable quality, we proceed to train the enhancement layer. In this phase, we freeze the base layer. We first train the merging-map generator and refinement module inside MFMN with $L = D(x'_t, x_t)$. Then, we train the MFMN together with the enhancement-layer CANF compressor without the skip-mask generator network. The loss function is $L = D(\hat{x}_t, x_t) + R_b + R_e$, where R_e is the estimated coding bitrate of



(a) M_m^1 (b) M_m^2 (c) M_m^3 (d) $M_m^1 * \hat{x}_t^{SR}$ (e) x_t'

Figure 4. Visualization of intermediate results produced by multi-frame merging network (MFMN). The top row (Jockey) has fast moving background and the bottom row (HoneyBee) has slow moving background.

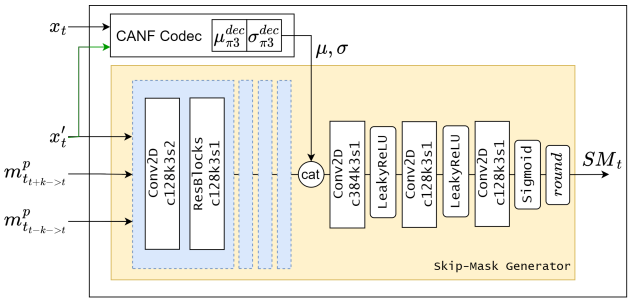
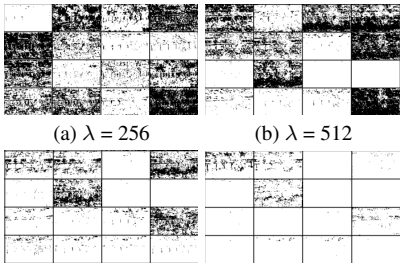


Figure 5. The skip-mask generator consists of convolutional layers and residual blocks. We use a sigmoid function to scale the output to a range between 0 and 1, followed by using a rounding operator to create a binary map.



(a) λ = 256 (b) λ = 512 (c) λ = 1024 (d) λ = 2048

Figure 6. Examples of the skip mask at different λ's (smaller λ's result in lower bitrates) for the model trained with mean-squared error (MSE). The transmitted latent variables have 128 channels; only 16 are shown for each bitrate.

the enhancement-layer CANF. When the above training step converges, we include the skip-mask generator network and activate the skip-mode coding process inside the adaptive CANF compressor for training.

4. In the final phase, we train all the modules in an end-to-end manner. We append Aux at the end of the loss function and introduce a parameter ε in front of R_b. Aux refers to $(D(y_2, x_t') + D(x_t', x_t) + D(\hat{x}_t^{SR}, x_t)) * 0.01 * \lambda$. It functions as a regularizer for y_2, x_t', and $\hat{x}_t^{SR}$. y_2 is the approximation of conditioning signal x_t' from CANF codec [10]. The parameter 0.01 is recommended by [15] although our terms are slightly different. Thus, the final loss function is $L = D(\hat{x}_t, x_t) * \lambda + \varepsilon * R_b + R_e + Aux$.

In total, we use five epochs to train RIFE with the initial model from [19], five epochs to train the base layer, five epochs to train the enhancement layer, and 25 epochs to train all the modules in an end-to-end manner. We reduce the learning rate when the loss function reaches a plateau. To obtain models for different bitrates, we choose λ = 256, 512, 1024, 2048 for training the mean-squared error (MSE) model and λ = 4, 8, 16, 32 for training the multi-scale structural similarity index (MS-SSIM) model. The MSE model adopts MSE as the distortion measure $D(\cdot, \cdot)$, and the MS-SSIM model adopts MS-SSIM. The ε parameter controls the bitrate (and thus image quality) of the base layer. In our experiment, ε = 4 is chosen empirically. To generate different rate points, we first train the model for the highest rate point (λ = 2048) using the complete training procedure and then fine-tune (phase 4 only) the resulting model for the other rates for five epochs.

4. Experiments

4.1. Dataset

The Vimeo90K septuplet dataset [33] was used to train our proposed method. It contains 91,701 7-frame sequences of resolution 448x256. During training, we randomly crop each frame to 256x256 and flip it horizontally and vertically. We evaluate our training models using the popular video coding test datasets: UVG [28] (7 videos) and HEVC Class B [7] (5 videos). The performance metrics are peak-signal-to-noise ratio (PSNR) and multi-scale structural similarity index (MS-SSIM) at several coding bitrates. We also calculate the BD-rate savings [6].

4.2. Rate-Distortion (RD) Performance

Figure 7 shows the RD performance on the test datasets using GOP=32. Our proposed method is denoted as TLZMC (Two-Layer Zero Motion Coding). When the FA technique is used, our method is denoted as TLZMC*. More results with different downsampling and super-

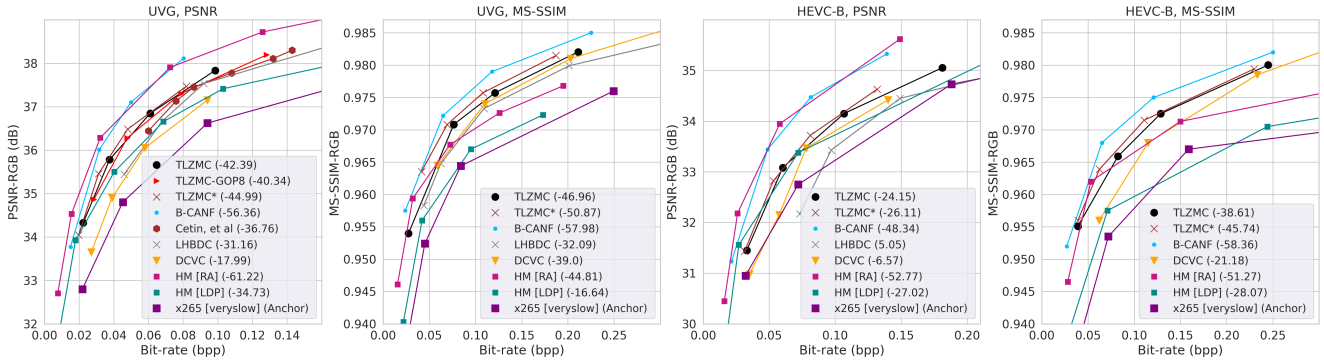


Figure 7. RD results (GOP=32) on UVG and HEVC Class B datasets measured in PSNR and MS-SSIM over bitrate (bpp). There are different evaluation settings for B-CANF (intra period=32, GOP=16), Çetin et al. [8] (GOP=16), and LHBDC (GOP=8).

resolution methods are provided in the supplementary document. Clearly, the RD performance of TLZMC* is somewhat better than that of TLZMC. Our methods are compared with DCVC [22] (a conditional P-frame coding scheme) and the other hierarchical B-frame coding methods: LHBDC [35] (GOP=8), Çetin et al. [8] (GOP=16), and B-CANF [10] (hybrid-based coding with intra period=32 and GOP=16), which is the state-of-the-art B-frame coding scheme. For classical coding, we include the RD curves of HM 16.23 [3] with *encoder_lowdelay_P_main* configuration (LDP) and with *encoder_randomaccess_main* configuration (RA / Random Access) and x265 [1] with *veryslow* mode (zerolatency). The BD-rate saving in the parenthesis is calculated using x265 (veryslow) with GOP=32 as anchor. We perform coding on all available frames for the UVG dataset, but only on the first 100 frames for the HEVC Class B dataset.

Except for B-CANF, our method outperforms all the other deep video codecs in PSNR. It should be noted that the B-CANF performance is based on an intra period of 32 and a GOP of 16 using its *B*-frame* technique. In comparison with the classical codecs, our method outperforms HM (LDP) and x265 (veryslow) but is lower than HM (RA). Regarding MS-SSIM, our method is slightly lower than B-CANF but outperforms the other deep video codecs and the classical codecs. It is interesting to observe that the performance of our method is closer to that of B-CANF at lower bitrates in MSE and MS-SSIM models.

Table 1 presents the bit distribution between the base and enhancement layers. Generally, the base layer consumes less than 7% of the total bitrate in average. However, when employing frame-type adaptive coding, the base-layer bitrate exhibits increased flexibility, reaching up to 18% in reference frames and 16% in non-reference frames.

4.3. Skip-Mask and Skip-Mask Generator

We show the benefit of our skip mask by calculating the percentage of retained latent samples (transmitted samples) at various bitrates (λ values). Table 2 shows the statistics of

λ	TLZMC	TLZMC*
	R / NR / AVG	R / NR / AVG
256	5.38 / 8.71 / 6.80%	18.05 / 15.96 / 17.41%
512	4.30 / 7.55 / 5.53%	11.58 / 8.45 / 10.52%
1024	2.99 / 6.43 / 4.51%	8.08 / 5.16 / 7.01%
2048	2.55 / 5.29 / 3.46%	5.31 / 2.83 / 4.31%
Average	3.80 / 6.99 / 5.07%	10.75 / 8.10 / 9.81%

Table 1. Percentages of the base-layer bit rate for 100 frames in all videos in the HEVC-B dataset. The percentages of the enhancement-layer bitrate can be derived by (100 - BL)%. The total bitrate excludes intra frames. R: reference frames, NR: non-reference frames, and AVG: the average percentages of the base-layer bitrate over both reference and non-reference frames.

retained samples using the MSE model on the UVG dataset (GOP=8). The percentage of retained samples is lower at lower bitrates (lower λ values) because fewer bits can be used to send transmitted samples. The average retained rates on the UVG dataset for λ = 256, 512, 1024, and 2048 are 28.28%, 36.23%, 57.17%, and 69.93%, respectively.

The skip-mask generator has two sets of inputs: (1) predicted motion data and merged frames, and (2) μ and σ from the hyperprior. Table 3 shows how each input contributes to the BD-rate saving. The evaluation is performed on the UVG dataset with GOP=32, and each model is separately trained in an end-to-end manner (phase 4). The BD-rate

Sequence	λ			
	256	512	1024	2048
YachtRide	40.75	53.21	64.01	70.22
Bosphorus	23.79	34.01	43.21	58.63
ShakeNDry	29.10	39.43	55.64	75.77
ReadySteadyGo	39.65	44.23	56.16	64.85
Beauty	20.99	29.73	81.63	93.65
Jockey	35.03	38.77	65.13	67.45
HoneyBee	8.66	14.21	34.42	58.96
Average	28.28	36.23	57.17	69.93

Table 2. The percentages of retained (transmitted) latent samples at different λ values. Smaller λ's result in lower bitrates.

	Input	BD-Rate Saving
(i)	Without skip mask	-35.87
(ii)	x'_t, $m^p_{t_{t-k\to t}}$, $m^p_{t_{t+k\to t}}$	-38.12
(iii)	μ, σ	-39.74
(iv)	(ii) & (iii)	-42.39

Table 3. Ablation study of the inputs to the skip-mask generator tested on the UVG dataset (GOP=32).

saving of both sets of inputs is significantly better that of the individual sets alone.

4.4. GOP Size

To better understand the RD performance under different GOP settings, we include our RD performance on the UVG dataset using GOP=8 (TLZMC-GOP8) in Figure 7. As shown, a larger GOP size leads to a slightly higher BD-rate saving. When tested on the UVG dataset, our method with GOP=32 performs comparably to B-CANF at lower bitrates in terms of both PSNR and MS-SSIM.

4.5. Computational Complexity

The complexity of our method is shown in Table 4 in terms of model size, runtimes, and multiply-and-accumulate operations (MACs). The test is run on GTX 2080Ti with GOP=32 on the UVG dataset. The MACs is calculated when encoding the first B-frame in a GOP. The encoding and decoding runtimes are averaged on the first 100 frames of Beauty sequence (UVG dataset), following the setting of [10]. Our MAC number is extracted using PyTorch library fvcore [2]. Because of the use of the CANF compressor in the base and enhancement layers, our model size reaches 39.9M, which is approximately 1.5x larger than the others. However, the number of pixels to the base-layer compressor is one-sixteenth of the full image resolution, resulting in a significant reduction in MACs and runtimes. Particularly, our encoder has only a slightly larger amount of computation than the decoder, while the other schemes have much higher encoder computation. This is because our encoder does not need to perform extra motion estimation for motion coding. Notably, our method has the lowest encoding time and its decoding time is very close to that of LHBDC, which has the lowest decoding time. Our encoding and decoding MACs are also very competitive.

We present a breakdown analysis of the encoder's model size and MACs in Table 5. Clearly, the base-layer multi-frame merging network and the enhancement-layer adaptive CANF use more than 80% of calculations. They may be subjected to further study for reducing computation.

5. Conclusion

We propose a two-layer video compression framework without motion coding. It is different from the mainstream hybrid-based coding framework in which motion coding is an essential component.

One critical element making our scheme successful is that we introduce a low-bitrate base layer that conveys the locations and values of the unpredictable pixels. One significant advantage of the proposed scheme is its low computational complexity, particularly at the encoder. Compared to the state-of-the-art learned B-frame codec [10] with similar coding components, our scheme has an RD performance slightly lower at high bitrates and about the same at low bitrates. On the other hand, our approach uses only 55% MACs operations in encoding and 73% MACs in decoding. This is the first attempt at designing a two-layer video compression scheme without motion coding. When the multi-frame merging network is replaced by a frame synthesis, the RD performance can be further improved as described in the supplementary document. Hence, there is a good potential to further improve its performance by tuning the parameters and altering the network architecture.

Model	Size	Encode		Decode	
		Time	MACs	Time	MACs
DCVC	8M	7.70s	1.05M/px	28.97s	0.68M/px
LHBDC	23.5M	1.19s	1.94M/px	0.73s	1.12M/px
B-CANF	24M	1.69s	2.70M/px	1.09s	1.97M/px
TLZMC	39.9M	**0.87s**	**1.50M/px**	**0.76s**	**1.45M/px**

Table 4. Computational complexity comparison with DCVC [22] (P-frame coding), LHBDC [35] and B-CANF [10].

Modules	Size	Ratio	MACs	Ratio
Frame Interpolator				
RIFE	10.7M	26.94%	0.09M/px	5.76%
Base Layer				
CANF	12.6M	31.61%	0.06M/px	4.16%
DS	0.1M	0.01%	0.01M/px	0.01%
SR-Net	0.3M	0.76%	0.09M/px	6.63%
MFMN	1.5M	3.74%	0.43M/px	29.03%
Enhancement Layer				
Skip Mask	1.1M	2.67%	0.02M/px	1.33%
Ad. CANF	13.6M	34.27%	0.80M/px	53.08%
Total	39.9M		1.50M/px	

Table 5. A breakdown analysis of the model size and MACs for the encoder components (frame interpolator, base layer, enhancement layer).

6. Acknowledgement

This work is partially supported by MediaTek and the National Science and Technology Council, Taiwan (under Grant MOST 110-2221-E-A49 -065 and MOST 110-2634-F-A49-006). We would like to thank National Center for High-performance Computing (NCHC), Taiwan, for providing computational and storage resources for our experiments, and Mu-Jung Chen for his feedback and support.

References

[1] Ffmpeg, 2022. https://www.ffmpeg.org/. 7

[2] Fvcore, 2022. https://github.com/facebookresearch/fvcore. 8

[3] Hm reference software for hevc, 2022. https://vcgit.hhi.fraunhofer.de/jvet/HM/-/tree/HM-16.23/. 7

[4] Eirikur Agustsson, David Minnen, Nick Johnston, Johannes Balle, Sung Jin Hwang, and George Toderici. Scale-space flow for end-to-end optimized video compression. In *CVPR*, pages 8503–8512, 2020. 2

[5] David Alexandre, Hsueh-Ming Hang, and Wen-Hsiao Peng. Two-layer learning-based p-frame coding with super-resolution and content-adaptive conditional anf. In *Proceedings of the 4th ACM International Conference on Multimedia in Asia*, pages 1–7, 2022. 5

[6] Gisle Bjontegaard. Calculation of average psnr differences between rd-curves. *Document VCEG-M33*, 2001. 6

[7] Frank Bossen. Common test conditions and software reference configurations. *JCTVC-L1100*, 12(7), 2013. 6

[8] Eren Çetin, M Akın Yılmaz, and A Murat Tekalp. Flexible-rate learned hierarchical bi-directional video compression with motion refinement and frame-level bit allocation. In *ICIP*, pages 1206–1210. IEEE, 2022. 3, 7

[9] Meixu Chen, Todd Goodall, Anjul Patney, and Alan C Bovik. Learning to compress videos without computing motion. *Signal Processing: Image Communication*, 103:116633, 2022. 2

[10] Mu-Jung Chen, Yi-Hsin Chen, Peng-Yu Chen, Chih-Hsuan Lin, Yung-Han Ho, and Wen-Hsiao Peng. B-canf: Adaptive b-frame coding with conditional augmented normalizing flows. *arXiv:2209.01769v1*, 2022. 2, 3, 5, 6, 7, 8

[11] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learning image and video compression through spatial-temporal energy compaction. In *CVPR*, pages 10071–10080, 2019. 2

[12] Abdelaziz Djelouah, Joaquim Campos, Simone Schaub-Meyer, and Christopher Schroers. Neural inter-frame compression for video coding. In *ICCV*, pages 6421–6429, 2019. 3

[13] Runsen Feng, Yaojun Wu, Zongyu Guo, Zhizheng Zhang, and Zhibo Chen. Learned video compression with feature-level residuals. In *CVPR Workshop*, pages 120–121, 2020. 2

[14] Yung-Han Ho, Chih-Chun Chan, Wen-Hsiao Peng, Hsueh-Ming Hang, and Marek Domański. Anfic: Image compression using augmented normalizing flows. *IEEE OJCAS*, 2:613–626, 2021. 3, 5

[15] Yung-Han Ho, Chih-Peng Chang, Peng-Yu Chen, Alessandro Gnutti, and Wen-Hsiao Peng. Canf-vc: Conditional augmented normalizing flows for video compression. In *ECCV*, pages 207–223. Springer, 2022. 2, 3, 6

[16] Zhihao Hu, Zhenghao Chen, Dong Xu, Guo Lu, Wanli Ouyang, and Shuhang Gu. Improving deep video compression by resolution-adaptive flow coding. In *ECCV*, pages 193–209. Springer, 2020. 2

[17] Zhihao Hu, Guo Lu, Jinyang Guo, Shan Liu, Wei Jiang, and Dong Xu. Coarse-to-fine deep video coding with hyperprior-guided mode prediction. In *CVPR*, pages 5921–5930, 2022. 2

[18] Zhihao Hu, Guo Lu, and Dong Xu. Fvc: A new framework towards deep video compression in feature space. In *CVPR*, pages 1502–1511, 2021. 2

[19] Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. Real-time intermediate flow estimation for video frame interpolation. In *ECCV*, pages 624–642. Springer, 2022. 4, 5, 6

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015. 5

[21] Théo Ladune, Pierrick Philippe, Wassim Hamidouche, Lu Zhang, and Olivier Déforges. Conditional coding for flexible learned video compression. *ICLR*, 2021. 3

[22] Jiahao Li, Bin Li, and Yan Lu. Deep contextual video compression. *NeurIPS*, 34, 2021. 2, 3, 7, 8

[23] Jiahao Li, Bin Li, and Yan Lu. Hybrid spatial-temporal entropy modelling for neural video compression. In *ACM Multimedia*, 2022. 2, 3

[24] Jianping Lin, Dong Liu, Houqiang Li, and Feng Wu. M-lvc: Multiple frames prediction for learned video compression. In *CVPR*, pages 3546–3554, 2020. 2

[25] Guo Lu, Chunlei Cai, Xiaoyun Zhang, Li Chen, Wanli Ouyang, Dong Xu, and Zhiyong Gao. Content adaptive and error propagation aware deep video compression. In *ECCV*, pages 456–472. Springer, 2020. 2

[26] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In *CVPR*, pages 11006–11015, 2019. 2

[27] Guo Lu, Xiaoyun Zhang, Wanli Ouyang, Li Chen, Zhiyong Gao, and Dong Xu. An end-to-end learning framework for video compression. *IEEE TPAMI*, 43(10):3292–3308, 2020. 2

[28] Alexandre Mercat, Marko Viitanen, and Jarno Vanne. Uvg dataset: 50/120fps 4k sequences for video codec analysis and development. In *MMSys*, pages 297–302, 2020. 6

[29] Reza Pourreza and Taco Cohen. Extending neural p-frame codecs for b-frame coding. In *ICCV, pages=6680–6689, year=2021*. 3

[30] Fabio Rocca. Television bandwidth compression utilizing frame-to-frame correlation and movement compensation. In *Symposium on Picture Bandwidth Compression*. Massachusetts Institute of Technology, 1969. 1

[31] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE TCSVT*, 22(12):1649–1668, 2012. 1

[32] Chao-Yuan Wu, Nayan Singhal, and Philipp Krahenbuhl. Video compression through image interpolation. In *ECCV*, pages 416–431, 2018. 2

[33] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *IJCV*, 127(8):1106–1125, 2019. 6

[34] Ren Yang, Fabian Mentzer, Luc Van Gool, and Radu Timofte. Learning for video compression with hierarchical quality and recurrent enhancement. In *CVPR*, 2020. 3

[35] M Akın Yılmaz and A Murat Tekalp. End-to-end rate-distortion optimized learned hierarchical bi-directional video compression. *IEEE TIP*, 31:974–983, 2021. 3, 5, 7, 8

[36] Nannan Zou, Honglei Zhang, Francesco Cricri, Hamed R Tavakoli, Jani Lainema, Emre Aksu, Miska Hannuksela, and Esa Rahtu. Adaptation and attention for neural video coding. In *ISM*, pages 240–244, 2021. 2

[37] Nannan Zou, Honglei Zhang, Francesco Cricri, Hamed R Tavakoli, Jani Lainema, Emre Aksu, Miska Hannuksela, and Esa Rahtu. Learned video compression with intra-guided enhancement and implicit motion information. In *ICCV*, pages 1870–1874, 2021. 2