B-CANF: Adaptive B-Frame Coding With Conditional Augmented Normalizing Flows

Mu-Jung Chen, Yi-Hsin Chen, and Wen-Hsiao Peng, Senior Member, IEEE

Abstract—Over the past few years, learning-based video compression has become an active research area. However, most works focus on P-frame coding. Learned B-frame coding is under-explored and more challenging. This work introduces a novel B-frame coding framework, termed B-CANF, that exploits conditional augmented normalizing flows for B-frame coding. B-CANF additionally features two novel elements: frame-type adaptive coding and B*-frames. Our frame-type adaptive coding learns better bit allocation for hierarchical B-frame coding by dynamically adapting the feature distributions according to the B-frame type. Our B*-frames allow greater flexibility in specifying the group-of-pictures (GOP) structure by reusing the B-frame codec to mimic P-frame coding, without the need for an additional, separate P-frame codec. On commonly used datasets, B-CANF achieves the state-of-the-art compression performance as compared to the other learned B-frame codecs and shows comparable BD-rate results to HM-16.23 under the random access configuration with the group-of-pictures (GOP) structure, without the need for an additional, separate P-frame codec.

Index Terms—Neural video coding, conditional coding, B-frame coding.

I. INTRODUCTION

The great success of learned image compression has spurred a new wave of research and development for learned video compression. Most existing methods [1], [2], [3], [4], [5], [6], [7] address low-delay P-frame coding based on hybrid-based coding architecture, which comprises uni-directional temporal prediction followed by deep residual coding. Ladune et al. [8] recently show that residual coding is sub-optimal from the information-theoretic perspective, proposing to condition the variational autoencoder (VAE)-based deep compression on motion-compensated frames [8], [9] with the aim of reaching the lower conditional entropy.

In contrast to the rapid progress in learned P-frame coding, learned B-frame coding, which allows bidirectional referencing for higher coding efficiency, is an under-explored topic. The coding of B-frames usually involves frame interpolation with explicit motion modeling and coding. Residual or conditional coding may then follow to achieve inter-frame prediction. Table I dissects some notable B-frame coding schemes in terms of their strategies for motion and inter-frame coding, to give an overview of recent developments in this area. As shown, most of the existing approaches [10], [11], [12], [13] adopt residual-based inter-frame coding, while many still use intra motion coding (e.g. encoding optical flow maps as individual images). Notably, the first B-frame coding scheme [14] already adopts conditional inter-frame coding, although its motion coding is still an intra-based approach. Built on top of [8], the more recent VAE-based conditional coding framework [9] features both conditional motion and inter-frame coding. However, its compression performance is inferior to the two state-of-the-art methods [12], [13], both adopt residual-based inter-frame coding. We argue that the full potential of conditional coding is yet to be seen. How it is implemented can make a big difference in coding performance.

Inspired by our previous work [15], we propose a novel learned B-frame coding framework, termed B-CANF, that exploits conditional augmented normalizing flows (CANF) for B-frame coding. We choose CANF as the coding backbone because of its generality and reversibility. It is shown in [15] that CANF is able to achieve greater expressiveness by stacking multiple conditional VAE encoders and decoders.

B-CANF extends [15] and differs from most prior works (e.g. [1], [2], [3], [4], [5], [6], [7], [16], [17], [18]) in several significant ways:

- B-CANF addresses primarily B-frame coding rather than P-frame coding. Similar to [15], it adopts conditional motion and inter-frame coding. However, the way in which the motion and inter-frame predictors are formulated has been adapted to B-frame coding.
- B-CANF newly introduces frame-type adaptive coding that learns better bit allocation for hierarchical B-frame coding.
- B-CANF newly proposes a special type of B-frame, called B*-frame, to mimic P-frame coding. This tool feature allows greater flexibility in specifying the group-of-pictures (GOP) structure, without the need for an additional, separate P-frame codec.

The non-trivial application of these novel elements—namely, CANF, frame-adaptive coding, and B*-frame—to B-frame coding achieves higher compression performance. To our best knowledge, B-CANF is also the first learned B-frame coding scheme that shows comparable PSNR results to HM-16.23 under the random access configuration with...
intra-period 32 and GOP size 16. Furthermore, this work presents a comprehensive study on B-frame coding, the results of which are not reported in our previous work [15] for P-frame coding. These facts distinct this work from [15] and highlight its contributions to advancing B-frame coding.

It is to be noted that this work is an expanded version of our conference publication [19], which adopts CANF for B-frame coding but focuses specifically on YUV 4:2:0 content, as opposed to RGB content in this work. Moreover, this work includes extensive ablation studies and complexity analyses, which are not covered in [19].

The remainder of this paper is organized as follows: Section II reviews learned video compression and the basics of ANF-based image compression (ANFIC [20]). Section III elaborates the design of B-CANF. Section IV compares B-CANF with the state-of-the-art methods and presents ablation experimental results. Finally, we provide concluding remarks in Section V.

II. RELATED WORK

A. Learned P-Frame Coding

End-to-end learned video compression has recently attracted lots of attention. Most prior works [1], [3], [4], [5], [6], [7], [11], [16], [21], [22], [23], [24], [25] focus on low-delay, P-frame coding, where a target frame is coded based on information propagated from the past decoded frames. In common, they share a similar temporal predictive coding architecture to the conventional codecs [26], [27]. As such, improving temporal prediction is one of the central research themes. In this aspect, there have been schemes such as motion-compensation networks [1], scale-space warping [3], one-stage motion estimation [7], feature-domain warping [21], multi-hypothesis prediction [4], compound spatiotemporal representation [17], and recurrent autoencoding [22], [23]. Because temporal prediction often relies on explicit motion modeling, some research efforts are dedicated to reducing motion overhead by predictive motion coding [4], incremental flow map coding [6], resolution-adaptive flow coding [5], [25] and coarse-to-fine motion coding [25]. Other notable P-frame coding techniques include the temporal prior [7], [22] that leverages a recurrent neural network to propagate causal, temporal information for entropy coding, adaptive residual skip coding [24], [25], and transformer-based autoencoders [28].

More recently, conditional coding [8], [15], [29], [30] achieves a breakthrough in inter-frame coding. It conditions the inter-frame autoencoder on the motion-compensated frame in forming a non-linear prediction of the target frame, as opposed to subtracting the motion-compensated frame from the target frame for residual coding, which is shown to be sub-optimal from the information-theoretic perspective [8]. Lately, Mentzer et al. [31] take an interesting approach to conditional coding, utilizing transformers [32] to model the dependencies between the latents of video frames for entropy coding. It has the striking feature of not requiring motion coding and warping operations. Although its initial idea dates back to [14], research on conditional coding remains very active.

B. Learned B-Frame Coding

In comparison with learned P-frame coding, learned B-frame coding [9], [10], [11], [12], [13], [14], which is the main focus of this paper and targets higher coding efficiency by allowing temporal prediction from both the future and past decoded frames, is relatively under-explored. Its process normally involves frame interpolation with explicit motion modeling and coding, followed by inter-frame residual coding [10], [11], [12], [13] or conditional coding [9], [14]. Due to the needs for bi-directional temporal prediction, B-frame coding usually incurs more motion overhead than P-frame coding. VAE-based intra motion coding [10], [11], [14], which encodes (bi-directional) optical flow maps as individual images, is very common. To minimize motion overhead, several strategies are proposed. Yang et al. [11] reuse the motion of a future or a past reference frame in a way similar to the DIRECT mode in [33]. Yilmaz et al. [13] adopt motion residual coding, with the predicted flow maps derived from the two reference frames. Ladune et al. [9] introduce one-stage, conditional motion coding, where the flow maps are estimated directly from and coded conditionally on the two reference frames without the use of optical flow estimation networks. Pourreza and Cohen [12] reuse the P-frame codec for B-frame coding, sending for each B-frame only one flow map that characterizes the motion between the target B-frame and its predicted frame interpolated by a pre-trained network. Efforts have also been made to share networks for coding different types of frames, e.g. I-frame, P-frame and B-frame [9], [12].

Table I contrasts the differences between our B-CANF and some notable learned B-frame coding schemes. Extending from our previous work [15] for P-frame coding, this paper introduces a conditional coding framework that exploits CANF for B-frame coding. As opposed to the other conditional coding schemes [9], [14], our method applies conditional coding

| Publication | Motion Coding for B-frames | Inter-frame Coding for B-frames | Modules with Shared Param. |
|-------------|-----------------------------|---------------------------------|---------------------------|
| ECCV'18     | VAE-based Intra             | Cond. VAE-based                 | No                        |
| ICCV'19     | VAE-based Intra             | VAE-based Residual              | I, B                      |
| CVPR'20     | VAE-based Intra & DIRECT    | VAE-based Residual              | No                        |
| KLRC'21     | One-stage, Cond. VAE-based  | Cond. VAE-based                 | I, P, B                   |
| ICCV'21     | VAE-based Intra             | VAE-based Residual              | B, P                      |
| TIP'22      | VAE-based Residual          | VAE-based Residual              | No                        |
| B-CANF (Ours) | Cond. ANF-based            | Cond. ANF-based                 | B, B*                     |
Fig. 1. Illustration of (a) the unconditional ANF for image coding in ANFIC [20] and (b) the proposed conditional ANF for B-frame coding, where $M$ signals the B-frame type for frame-type adaptive coding.

to both motion and inter-frame coding. Moreover, it features frame-type adaptive coding, which adapts the coding process to the B-frame type. We also introduce a special type of B-frame, called B*-frame, which mimics P-frame coding by reusing the B-frame codec.

C. Augmented Normalizing Flows for Image Compression

B-CANF and our previous work [15] are a sequel to the augmented normalized flow (ANF)-based image compression, also known as ANFIC [20], which adopts ANF [34] as the compression backbone. To ease the understanding of our B-CANF, this section reviews the basics of ANFIC [20]. Fig. 1(a) illustrates its coding architecture. Shown on the left are the three inputs of ANFIC [20]: the image to be encoded $x$ and the two augmented inputs $e_1, e_2$; accordingly, the compressed outputs are a nearly zero image $y_2 \approx 0$, the quantized latent code $\hat{z}_2$, and the hyperprior $\hat{h}_2$.

1) Encoding: The encoding process proceeds from left to right. The input pair $(x, e_2)$ with $e_2 = 0^1$ is transformed into the output pair $(y_2, \hat{z}_2)$, where $y_2$ is regularized to approximate a zero image, by two autoencoding transforms $(A_1, S_1)$ and $(A_2, S_2)$ arranged in the form of additive coupling layers. $A_1, A_2$ are VAE-based analysis transforms, while $S_1, S_2$ are VAE-based synthesis transforms. In symbols, the intermediate states $y_1, z_1$ after the first autoencoding transform $(A_1, S_1)$ are given by

$A_1(x_1, e_2) = (x_1, e_2 + \mu_{A_1}(x_1)) = (x_1, z_1), \quad (1)$

$S_1(x_1, z_1) = (x_1 - \mu_{S_1}(z_1), z_1) = (y_1, z_1), \quad (2)$

where $\mu_{A_1}, \mu_{S_1}$ are two neural networks that output element-wise additive transform parameters. The second autoencoding transform $(A_2, S_2)$ that converts $y_1, z_1$ into $y_2, \hat{z}_2$ follows a similar definition, except that $z_2$ is additionally predicted from the hyperprior and quantized as $\hat{z}_2$. Here $\hat{H}_A, \hat{H}_S$ serve as the hyperprior autoencoder [35], which encodes the latent code $\hat{z}_2$ into $\hat{h}_2$ for entropy coding. The reader is referred to [20] for network details. By doing so, ANFIC transforms the information from $x_1$ into the augmented space $\hat{z}_2, \hat{h}_2$, which have much lower resolution than the original image and allow for more efficient coding.

2) Decoding: The decoding proceeds in reverse order by decoding first the hyperprior $\hat{h}_2$, followed by the decoding of $\hat{z}_2$ to update $y_2$ (initialized as 0) successively from right to left. During training, $e_2 \sim U(-0.5, 0.5)$ simulates the additive quantization noise of the hyperprior. Remarkably, the conventional VAE-based compression is a special case of ANFIC by retaining only the autoencoding transform $(A_2, S_2)$. In this sense, ANFIC is able to achieve superior expressiveness to VAE-based compression systems by cascading more autoencoding transforms.

Table II summarizes the differences of ANFIC [20], our previous work CANF-VC [15], and B-CANF. Both CANF-VC [15] and B-CANF adopt CANF as the coding backbone, whereas ANFIC uses ANF. CANF differs from ANF in that it learns conditional distributions of video frames rather than unconditional distributions of images (Section III-B). Moreover, B-CANF distinguishes from CANF-VC [15] by the support of B-frames and B*-frames (Section III-C); additionally, it incorporates frame-type adaptive coding (Section III-D). Extensive experimental results and ablation studies on B-frame coding reported in this paper are not seen in [15].

III. PROPOSED METHOD

A. System Overview

We begin this section with an overview of our B-frame coding framework. Specifically, we target the two common hierarchical Group-of-Pictures (GOP) prediction structures shown in Fig. 2, where Fig. 2(a) depicts the case with an intra-period of the same size as the GOP and Fig. 2(b) the other case having multiple GOPs in one intra-period. In the latter case, we introduce a special type of B-frame, known as B*-frame, the coding of which reuses the same framework as for regular B-frames (including both reference and non-reference ones). Our B*-frames are conceptually similar to

---

1ANF [34] is initially designed to be a generative model, where the augmented noises are meant to induce a complex marginal on the input $x_i$ and the entropy rate of the latents is not a concern. In ANFIC [20], the latents $z_2$ must be quantized and compressed to achieve low entropy. Injecting noise at $e_2$ will increase the entropy of $z_2$. ANFIC [20] shows that having $e_2 = 0$ can still achieve good compression performance.

---

Table II: Comparison of B-CANF, ANFIC [20], and CANF-VC [15]

| Codec Type | Frame-type Support | Frame-type Adaptive Coding |
|------------|-------------------|---------------------------|
| ANFIC [20] | ANF-based         | N                         |
| CANF-VC [15] | CANF-based | I, P | N |
| B-CANF (Ours) | CANF-based | I, B, B* | Y |

---
generalized B-frames [36], which allow the two reference frames to be both from the past (and even the same) decoded frames. The use of B*-frame is to mimic P-frame coding with our B-frame coding framework, an approach that is in direct contrast to [12], which uses P-frame coding for B-frames. We remark that B-frame coding is more flexible and general than P-frame coding.

Fig. 3(a) illustrates our B-frame coding framework. Consider the coding of a B-frame $x_t$, where the subscript indicates the time index, given its two reference frames $\hat{x}_{t-k}, \hat{x}_{t+k}$, i.e., the previous and subsequent decoded reference frames at higher hierarchical levels, respectively. Our first step is to interpolate a predicted frame $\hat{x}^c_t$ through bidirectional motion compensation in pixel domain. To this end, we estimate and signal two optical flow maps $m^p_{t\rightarrow t-k}, m^p_{t\rightarrow t+k}$ for bidirectional backward warping. Our main focus here is on reducing motion overhead through conditional coding. To obtain the conditioning signals, we introduce a motion prediction network that derives the (predicted) optical flow maps $m^p_{t\rightarrow t-k}, m^p_{t\rightarrow t+k}$ from observing the two reference frames $\hat{x}_{t-k}, \hat{x}_{t+k}$ without having access to $x_t$. Our second step is to code the B-frame $x_t$ efficiently by conditioning the inter-frame coding on the predicted frame $\hat{x}^c_t$. Our framework is comprised of the following major components:

- **Motion estimation network** (ME-Net) estimates the backward flow maps $m^e_{t\rightarrow t-k}, m^e_{t\rightarrow t+k}$ between $x_t$ and the two reference frames $\hat{x}_{t-k}, \hat{x}_{t+k}$, respectively. In this work, we adopt SPyNet [37] for motion estimation.

- **Motion prediction network** predicts two optical flow maps $m^p_{t\rightarrow t-k}, m^p_{t\rightarrow t+k}$, which serve jointly as thecondition for coding $m^e_{t\rightarrow t-k}, m^e_{t\rightarrow t+k}$, from observing the two reference frames $\hat{x}_{t-k}, \hat{x}_{t+k}$ without having access to $x_t$. We follow [38] in using a convolutional neural network to generate $m^p_{t\rightarrow t-k}, m^p_{t\rightarrow t+k}$ in a coarse-to-fine fashion.

- **CANF-based motion codec**, conditioned on the predicted flow maps $m^p_{t\rightarrow t-k}, m^p_{t\rightarrow t+k}$, encodes the flow maps $m^e_{t\rightarrow t-k}, m^e_{t\rightarrow t+k}$ jointly to output the reconstructed flow maps $\hat{m}_{t\rightarrow t-k}, \hat{m}_{t\rightarrow t+k}$ (Section III-B).

- **Frame synthesis network** synthesizes the predicted frame $\hat{x}^c_t$ by GridNet [39], which takes as inputs the motion-compensated reference frames and their features generated based on the reconstructed flows $\hat{m}_{t\rightarrow t-k}, \hat{m}_{t\rightarrow t+k}$.

- **CANF-based inter-frame codec** encodes the target frame $x_t$, conditioned on the predicted frame $\hat{x}^c_t$ (Section III-B).

### B. CANF-Based Inter-Frame and Motion Codecs

Fig. 1(b) presents the architecture of our CANF-based codec. Both our inter-frame and motion codecs adopt a similar design. To keep the notation uncluttered, we take the inter-frame codec as an illustrative example.

**CANF-based inter-frame codec** seeks to learn the conditional distribution $p(x_t | \hat{x}^c_t)$ of the B-frame $x_t$, given the bidirectionally predicted frame $\hat{x}^c_t$. As compared to ANFIC [20], which employs an unconditional ANF to learn the image distribution $p(x_t)$ for image compression, our CANF-based inter-frame codec introduces two novel elements: (1) conditional autoencoding transforms and (2) conditional entropy coding with the combined hyperprior and temporal prior.

From Fig. 1(b), the conditional autoencoding transform is designed to transform the B-frame $x_t$ into the latent $y_2$, which in the present case is regularized to approximate the predicted frame $\hat{x}^c_t$ instead of a zero image as with ANFIC [20] (cp. Fig. 1(a)). In the course of transformation, the latent $\hat{y}_2$ encodes the information necessary for instructing the transformation from $x_t$ to $\hat{x}^c_t$ and vice versa, while the hyperprior $\hat{h}_2$ is utilized to model the distribution of $\hat{y}_2$ for entropy coding. Taking $\{A_1, S_1\}$ in Fig. 1(b) as an example, we have

$$A_1(x_t, e_2 | \hat{x}^c_t) = (x_t, e_2 + \mu_{A_1}(x_t, \hat{x}^c_t)) = (x_t, z_1),$$

$$S_1(x_t, z_1) = (x_t - \mu_{S_1}(z_1), z_1) = (y_1, z_1),$$

where $\mu_{A_1}, \mu_{E_2}$ are two neural networks that output element-wise additive transform parameters. It is worth noting that the encoding process $\mu_{A_1}$ is conditioned on $\hat{x}^c_t$ by concatenating $\hat{x}^c_t, x_t$ to form the encoder input. The rationale behind the design is to ease the transformation from $x_t$ to $\hat{x}^c_t$ by supplying the target $\hat{x}^c_t$ as an auxiliary signal. This process is repeated by taking the resulting $y_1, z_1$ as inputs to the next autoencoding transform $\{A_2, S_2\}$.

The conditional entropy coding is achieved by the hierarchical autoencoding transform $(H_a, H_s)$, which estimates the conditional distribution of $z_2$ given the hyperprior $\hat{h}_2$ and the interpolated frame $\hat{x}^c_t$ for entropy coding. Its operation is governed by

$$H_a(z_2, e_h) = (z_2, e_h + \mu_{H_a}(z_2)) = (z_2, \hat{h}_2),$$

$$H_s(z_2, \hat{h}_2 | \hat{x}^c_t) = (z_2 - \mu_{H_s}(\hat{h}_2, H_p(\hat{x}^c_t)), \hat{h}_2) = (\hat{z}_2, \hat{h}_2),$$

where $[\cdot]$ (depicted as Q in Fig. 1(b)) denotes the nearest-integer rounding at inference time for coding $z_2$ in a lossy manner. In particular, $\hat{h}_2 = e_h + \mu_{H_a}(z_2)$ denotes the quantized hyperprior and is entropy coded by a learned factorized distribution [40], with $e_h \sim U(-0.5, 0.5)$ simulating additive quantization noise applied to the hyperprior latent $\mu_{H_a}(z_2)$. $\mu_{H_s}(\hat{h}_2, H_p(\hat{x}^c_t))$ models the mean of the latent $z_2$, which is assumed to follow a Gaussian distribution with the standard deviation determined by another output $\sigma(\hat{h}_2, H_p(\hat{x}^c_t))$ tied to the same backbone network as $\mu_{H_a}$. We remark that the combination of $\hat{z}_2, \hat{x}^c_t$ in deriving the coding probabilities for $z_2$ exerts a combined effect of the hyperprior and temporal prior.

In Fig. 1(b), another conditioning factor $M$ is utilized to signal the frame type of $x_t$—namely, reference B-frames,
non-reference B-frames, or B*-frames—and to adapt feature distributions for frame-type adaptive coding (Section III-D).

**CANF-based motion codec** follows a similar design to our CANF-based inter-frame codec. The changes include (1) replacing the coding B-frame $x_t$ with the concatenation of the two optical flow maps $\{m^e_{t \rightarrow t-k}, m^e_{t \rightarrow t+k}\}$ to be coded, and (2) replacing the interpolated frame $\hat{x}_{t-k}$ with the concatenation of the two predicted optical flow maps $\{m^p_{t \rightarrow t-k}, m^p_{t \rightarrow t+k}\}$. It is important to note that $\{m^p_{t \rightarrow t-k}, m^p_{t \rightarrow t+k}\}$ are coded jointly based on the predicted flows $\{m^p_{t \rightarrow t-k}, m^p_{t \rightarrow t+k}\}$, allowing the motion codec to exploit correlations between these flow maps for reducing motion overhead.

The major differences between the CANF-based codecs of our B-CANF and [15] are summarized as follows:

- **Bidirectional joint motion coding**: B-CANF addresses B-frame coding. There are two optical flow maps (rather than one in [15]) to be coded. Instead of encoding these flow maps independently, they are concatenated as the input to our CANF-based motion codec for joint coding. Likewise, the motion prediction module outputs two flow map predictors. They are also concatenated as the joint conditioning signal. This allows our CANF-based motion codec to best use the contextual information from both flow map predictors for joint coding. As a result, the channels of the input, conditional input, and output are twice as many as those of their counterparts in [15].

- **Frame-type adaptive coding**: B-CANF additionally incorporates a frame-type adaption module in every convolutional layer of the CANF-based motion and inter-frame codecs, in order to adapt the feature distributions for frame-type adaptive coding. We will discuss this in Section III-D.

- **Network configuration**: The network settings, such as the number of input/output channels, kernel size, and stride, for the CANF-based motion and inter-frame codecs have been changed in this work, in order to strike a better balance among coding efficiency, model size, and computational complexity. More details will be provided in the released code.

### C. B*-Frame Extension

B*-frames allow to support multiple GOPs in an intra-period as depicted in Fig. 2(b), to strike a balance between coding performance and latency. A straightforward approach is to encode P-frames periodically using an additional P-frame codec. In doing so, however, the total model size is increased. To address this issue, we propose B*-frames, which reuse our CANF-based B-frame codec to mimic P-frame coding.

To encode a video frame $x_t$ in B* mode, we create a predicted frame from the past decoded frame $x_{t-k}$ just like coding a vanilla P-frame. To this end, the flow map $m^e_{t \rightarrow t-k}$
between \(x_t\) and \(x_{t-k}\) is estimated and coded. As shown in Fig. 3(b), the main inputs to our conditional motion codec should ideally comprise two optical flow maps. We generate the other “virtual” input by flipping the sign of \(m_{t\rightarrow t-k}^r\) (i.e. \(-1 \times m_{t\rightarrow t-k}^r\)) in order to match the characteristics of \(m_{t\rightarrow t-k}^r\) (for ordinary B-frame coding). Because only one reference frame \(x_{t-k}\) is used, we disable motion prediction network and set the predicted flow maps to constant 0. This reduces our motion coding scheme for B*-frames to intra-based motion coding. Although it is possible to use a motion extrapolation network to generate a flow map predictor from previously coded B*-frames, this calls for long-term motion extrapolation. It is to be noted that B*-frames are usually temporally distant from each other or may even be coded only once in an intra-period as with the HM randomaccess configuration [41].

The predicted frame \(\hat{x}_t^r\) is then synthesized from the reference frames based on the coded \(m_{t\rightarrow t-k}^r\) and its virtual counterpart (after sign reversal), which are found empirically to be similar but not exactly the same (see ablation experiments in Section IV-D). As such, the B*-frame is by definition a bi-prediction frame.

Our ablation experiments in Section IV-D show that B*-frames have similar compression performance to P-frames, which require an additional, separate P-frame codec.

### D. Frame-Type Adaptive Coding

Frame-type adaptive coding aims to achieve adaptive coding according to the reference types of B-frames. In traditional codecs, the reference B-frames are usually coded at higher quality than the non-reference B-frames by operating the same B-frame codec in different modes. Following a similar strategy, we weight more heavily the distortions of the reference B-frames and B*-frames during training (Section III-E). In addition, we observe that using a fixed, shared B-frame codec for coding all the types of B-frames is unable to achieve frame-type adaptive coding. On the other hand, training separate models for different B-frames as with [14] is prohibitively expensive.

We tackle this issue by introducing a frame-type adaptation (FA) module to every convolutional layer, aiming to share the convolutional layers between B-frames while adapting their features to the B-frame type. Specifically, according to the B-frame type (namely, reference B-frames, non-reference B-frames, or B*-frames), our FA module applies a channel-wise affine transformation to the output features of every convolutional layer. Take as an example the analysis transform in Fig. 4. The FA module is inserted between the convolutional layer and the generalized divisive normalization (GDN) layer. It takes B-frame type \(M\) (in one-hot vector) as input and generates the affine transform parameters \(\gamma\) and \(\beta\) for feature \(F\) adaptation by

\[
\text{FA}(F|M) = \gamma(M) \odot F + \beta(M)
\]

where \(\odot\) is channel-wise multiplication and addition. The FA modules appear in the autoencoding transforms and the hyperprior models. They are trained end-to-end together with the other networks. Note that similar feature adaptation techniques are widely used in computer vision tasks [42], [43] and are also adopted in end-to-end learned image compression [44]. Unlike these prior works, which take a set of feature maps as inputs to perform element-wise affine transformation, our FA module performs channel-wise affine transformation to adapt feature distributions with respect to a scalar indicating the input frame’s type.

Section IV-D shows that frame-type adaptive coding improves bit allocation because our B-frame codec is able to adapt feature distributions on a frame-by-frame basis. In passing, this tool feature is to be distinguished from training separate models for different rate points, which is still the case in all the experiments.

### E. Training

The model training is done on 5-frame training sequences (with their frames denoted by \(x_0, x_1, \ldots, x_4\)). In coding every training sequence, we follow the GOP-1 structure in Fig. 2(b), which encodes \(x_0\) as an I-frame by a pre-trained image codec [20] (not included for training), \(x_4\) as a B*-frame, and \(x_2\) as a reference B-frame. Moreover, we randomly choose between \(x_1\) and \(x_3\) for coding, in order to have a balanced distribution over B-frame types. Assuming that \(x_3\) is chosen, we formulate our training objective as follows:

\[
L = \lambda_1 \times D + R + \lambda_2 \times F,
\]

where

\[
D = d(x_4, \hat{x}_4) + \frac{1}{\alpha_r} d(x_2, \hat{x}_2) + \frac{1}{\alpha_{nr}} d(x_3, \hat{x}_3),
\]

\[
R = \sum_{i=2}^4 r(\hat{x}_i),
\]

\[
F = f_3 + \frac{1}{\alpha_r} f_2 + \frac{1}{\alpha_{nr}} f_3,
\]

with \(D\) denoting the weighted sum of frame distortions \(\{d(x_i, \hat{x}_i)\}_{i=2}^4\), \(R\) the accumulated bits \(\{r(\hat{x}_i)\}_{i=2}^4\) consumed by both the conditional motion and inter-frame coding, and \(F\) the sum of regularization losses \(\{f_i\}_{i=2}^4\) requiring that \(y_2\) in Fig. 1 (b) should approximate their respective conditions, i.e. \(\{\hat{x}_i^c\}_{i=2}^4\). This regularization loss is to have the latent representations \(\hat{z}_2, \hat{h}_2\) (Fig. 1 (b)) capture the information needed to signal the transformation between
IV. EXPERIMENTS

A. Settings

a) Training Details: For training, we use Vimeo-90k septuplet dataset [45], which includes 91,701 7-frame sequences of size 448 × 256. The training sequences are randomly cropped to 256 × 256, and five consecutive frames are selected. We adopt Adam [46] optimizer with learning rate $10^{-4}$ and batch size 16. To optimize our models for PSNR-RGB, we choose $\lambda_2 = 0.01\lambda_1$ and $\lambda_1 = 128, 512, 1024, 2048$ to train separate models for different rate points; likewise, to optimize them for MS-SSIM-RGB, we choose $\lambda_2 = 0.01\lambda_1$ and $\lambda_1 = 2, 8, 16, 32$. For all the experiments, $\alpha_r = 1, \alpha_n = 2$. These hyper-parameters are chosen to weight more heavily the quality of reference B-frames (including B*-frames).

b) Evaluation Methodologies: We evaluate our method on UVG [49], MCL-JCV [50], and HEVC Class B [51]. In addition, we also evaluate on CLIC’22 test dataset [52], which contains some challenging sequences with characteristics rarely seen in Vimeo-90k training dataset [45]. To compare fairly with HM (HEVC Test Model Version 16.23) [41] under the common test conditions, we follow the coding structure of the encoder_randomaccess_main configuration. That is, the intra-period is 32 and GOP size is 16 for all the test sequences. For every test sequence, we encode most of the video frames to the extent that the coded sequence contains the maximum number of intra-periods (32). As an example, for a 600-frame test sequence, we will encode 577 frames (1 I-frame + 18 intra-frames). Note that the source videos are in YUV420 format. The traditional codecs operate internally in YUV420, whereas the learned codecs perform YUV420-to-RGB444 conversion prior to coding. The reconstructed quality is measured in PSNR-RGB and MS-SSIM-RGB, and the bit-rate in bits-per-pixel (bpp). Table III summarizes the settings of these competing methods.

c) Baseline Methods: The baseline methods include traditional codecs and learning-based methods. The traditional codecs are x265 [53] in vseryslow mode (zerolateness), HM [41] with the encoder_lowdelay_P_main and encoder_randomaccess_main configurations, and VTM [54] with the encoder_randomaccess_main configuration. The learning-based methods include CANF-VC [15], DCVC [29] and two concurrent works, Sheng et al. [30] and Li et al. [48], which are the state-of-the-art learned P-frame coding schemes. We also compare B-CANF with LHDBC [13], the state-of-the-art learned B-frame coding scheme. To align the I-frame codec, we additionally evaluate their performance using ANFIC [20] as the I-frame codec. We refer to these variants as DCVC (ANFIC), Sheng et al. (ANFIC), Li et al. (ANFIC) and LHDBC (ANFIC), respectively. Results for the baselines and their variants are produced with the code released by the authors.

B. Rate-Distortion Performance

When comparing our B-CANF with CANF-VC [15] and LHDBC [13], we observe that B-CANF surpasses them in terms of both PSNR-RGB and MS-SSIM-RGB across all the test datasets. However, B-CANF falls short of achieving the same level of performance as Li et al. [48], which is considered the state-of-the-art learned P-frame codec. This discrepancy in performance can be attributed to the more advanced entropy coding model employed by Li et al. [48] and the domain shift issue that arises between the training and testing phases. For a comprehensive analysis of this domain shift, please refer to Section IV-C.

Table V further compares our method with HLVC [11], RLVC [22], LHDBC [13], and B-EPIC [12] following their suggested settings. All the baseline methods adopt or support B-frame coding. In this experiment, these baseline methods are used as anchors for BD-rate evaluation. In addition, we note that the codes for B-EPIC [12] are unavailable. We thus copy the results from their paper in evaluating BD-rates.

In HLVC [11], a three-level hierarchical coding structure with P- and B-frames is used for encoding GOP’s of size 10. We follow exactly the same coding structure as HLVC [11] to perform encoding with B-CANF, except that we use B*-frames to replace P-frames. Recall that B*-frames are used to mimic P-frames. In RLVC [22], a bidirectional P-frame coding structure is proposed to take advantage of the past and future frames for better coding efficiency. For the same purpose, our B-CANF proposes B-frame coding. To demonstrate the best achievable performance of B-CANF, we adopt hierarchical B-frame coding for encoding GOP’s of size 13 in comparison with RLVC [22]. In LHDBC [13], a hierarchical B prediction structure with GOP 8 and intra-period 8 is proposed. In comparison with B-EPIC [12], which adopts a GOP size of 12 with an infinite intra-period (only one I-frame in the beginning), our B-CANF uses the same GOP size. However, we set the intra-period to 12 (instead of infinity) to mitigate the temporal error propagation. Despite this leads to more frequent coding of I-frames, its advantages outweigh the temporal error propagation. Under these various settings, our B-CANF (which adopts ANFIC [20] as the I-frame codec) still achieves considerable BD-rate reductions over HLVC [11], RLVC [22], LHDBC [13], and B-EPIC [12]. This is evidenced by the negative BD-rate numbers in the upper half of Table V.

C. Domain Shift in Training B-Frame Coders

The performance gap between our B-CANF and the works of Sheng et al. [30] and Li et al. [48] deserves further investigation. A crucial aspect that is often overlooked when comparing the compression performance of learned P-frame and B-frame codecs is the training datasets. Most learned codecs (P-frame and B-frame) are trained on Vimeo dataset [45], which consists of short video sequences, each containing only 7 frames. Consequently, we use 5-frame GOPs with 3-level B-frame coding for training. Notably, these training B-frames are temporally adjacent to each other.

However, during testing, we employ a large GOP of size 32 (i.e. the intra-period is 32) with 5 hierarchical temporal
TABLE III

Settings of the Competing Methods

| I-frame codec        | Coding structure | Intra-period | GOP |
|----------------------|------------------|--------------|-----|
| x265 (veryslow)     | -                | -            | 32  |
| HM (LDP)             | Hierarchical P   | 32           | 8   |
| HM (randomaccess)    | Hierarchical B   | 32           | 16  |
| VTM (randomaccess)   | Hierarchical B   | 32           | 32  |
| DCVC [29]            | cheng2020 [47]/hyperprior [35] | 32           | 32  |
| DCVC (ANFIC)         | ANFIC [20]/ANFIC | 32           | 32  |
| CANF-VC [15]         | ANFIC/ANFIC     | 32           | 32  |
| Sheng 22 [30]        | cheng2020 [47]/cheng2020 | 32           | 32  |
| Li’22 (ANFIC)        | ANFIC [20]/ANFIC | 32           | 32  |
| Li’22 (ANFIC)        | ANFIC [20]/ANFIC | 32           | 32  |
| LHBDC [13]           | mb2018-mean [47]/- | 32           | 16  |
| B-CANF (Ours)        | ANFIC/ANFIC     | 32           | 16  |

BD-RATE COMPARISON WITH INTRA-PERIOD 32. THE ANCHOR IS X265 IN VERYSLOW MODE. THE BEST PERFORMER IS MARKED IN RED AND THE SECOND BEST IN BLUE.

| BD-rate (%) PSNR-RGB | BD-rate (%) MS-SSIM-RGB |
|----------------------|------------------------|
|                      | UVG | MCL-JCV | HEVC-B | CLIC’22 | UVG | MCL-JCV | HEVC-B | CLIC’22 |
| HM (LDP)             | -30.6 | -26.4 | -23.0 | -16.8 | -21.6 | -17.3 | -18.0 | -8.8 |
| HM (randomaccess)    | -53.6 | -48.2 | -51.4 | -41.3 | -46.3 | -40.8 | -53.3 | -39.2 |
| VTM (randomaccess)   | -68.2 | -63.6 | -66.4 | -54.6 | -69.6 | -65.6 | -71.1 | -60.9 |
| DCVC [29]            | -2.3  | -7.6   | -3.0  | 28.7  | -45.2 | -9.3   | -55.6 | -25.9 |
| DCVC (ANFIC)         | -3.6  | -9.0   | -2.3  | 25.5  | -47.1 | -51.6  | -55.8 | -29.0 |
| CANF-VC [15]         | 28.7  | -21.3  | -21.4 | 3.3   | -49.8 | -50.4  | -55.0 | -30.3 |
| Sheng’22 [30]        | 45.7  | -38.0  | -43.1 | -9.5  | -63.6 | -63.8  | -73.9 | -58.2 |
| Sheng’22 (ANFIC)     | 41.8  | -38.0  | -40.9 | -9.9  | -63.0 | -64.7  | -73.3 | -57.1 |
| Li’22 (ANFIC)        | -62.4 | -54.0  | -57.6 | -26.9 | -73.0 | -72.8  | -80.4 | -67.9 |
| Li’22 (ANFIC)        | -59.2 | -51.2  | -54.7 | -26.3 | -72.1 | -71.9  | -79.6 | -66.6 |
| LHBDC [13]           | -18.6 | -5.3   | -4.4  | 38.8  | NA    | NA     | NA    | NA     |
| LHBDC (ANFIC)        | -18.6 | -6.7   | -3.4  | 38.2  | NA    | NA     | NA    | NA     |
| B-CANF (Ours)        | -47.5 | -41.1  | -46.9 | -33.8 | -60.2 | -61.2  | -67.1 | -59.6 |

Fig. 5. Rate-distortion plots on UVG, MCL-JCV, HEVC Class B and CLIC’22 test dataset in terms of PSNR-RGB and MS-SSIM-RGB.

levels in most cases. This results in a substantial number of B-frames being predicted from distant reference frames, leading to a significant domain shift (i.e. a significant change in statistics) between the training and test scenarios. This discrepancy can negatively affect the generalization of our ME-Net, motion prediction network, and frame synthesis network, especially when dealing with videos featuring substantial motion. In contrast, learned P-frame codecs like Li et al. [48] and Sheng et al. [30] have more consistent training and test scenarios in terms of temporal prediction distance. In P-frame
coding, the domain shift can occur after coding several frames because the quality of the reference frame often degrades over time. In B-frame coding, it happens right from the first coding B-frame, affecting all the subsequent frames. This early domain shift in B-frame coding has a more immediate and pronounced impact on compression performance.

To mitigate the domain shift between the training and test scenarios, we present additional results in Table VI for a smaller GOP size of 8. In order to ensure a fair comparison, we utilize ANFIC as the I-frame codec for all the competing methods. The results demonstrate that our B-CANF performs comparably to or better than Li et al. [48] under GOP=8, even without utilizing advanced entropy coding. Note that these results are not intended to advocate the use of small GOPs or short intra-periods. Rather, they serve to highlight the limitations of Vimeo dataset [45] in training B-frame codecs and demonstrate the potential of B-CANF.

### D. Ablation Experiments

We conduct ablation experiments with GOP 16 and intra-period 32. Unless otherwise specified, the BD-rates are reported against x265 (veryslow) in terms of PSNR-RGB.

1) **Conditional vs. Residual Coding:** Table VII analyzes the impact of our conditional coding scheme on compression performance. We test four variants, switching between residual and conditional coding for motion and inter-frame coding. When residual coding is selected, we use ANFIC [20] to code flow map or frame prediction residuals. Comparing variants A and D, turning both motion and inter-frame codecs into residual-type ones results in a considerable performance drop. In particular, our conditional coding scheme is seen to benefit inter-frame coding more than motion coding (from D to B vs. from D to C). A breakdown analysis of the bitstream composition explains that the motion part represents only 5-15% of the entire bitstream. Interestingly, the gains of our conditional coding in improving motion coding and inter-frame coding are nearly additive.

2) **Frame-type Adaptive Coding:** As mentioned in Section III-D, there are two mechanisms that contribute to the better bit allocation of our B-CANF. The first is to weight differently the distortions of different types of B-frames during training. The second is the FA module, which allows both the conditional motion and inter-frame codecs to behave differently according to the input frame’s type. Table VIII presents an ablation study to shed lights on how they contribute to the rate-distortion performance. Comparing setting A (our B-CANF) with setting B, we see that turning off the FA module leads to a significant rate-distortion performance loss. In this case, the same (amortized/average) motion and inter-frame codecs are used to process different types of B-frames. Likewise, in comparison with the full model (setting A), having only the FA module without weighting the distortions according to the frame type (setting C) during training is sub-optimal too. These results suggest that they both contribute significantly to the resulting rate-distortion performance.

Fig. 6 further visualizes how the proposed FA module may impact the bit allocation among B-frames in a GOP and their decoded quality. In this and the following experiments, the distortion weighting is enabled during training. We see that disabling the FA module (i.e., setting B in Table VIII) will learn an “average” codec that tends to allocate more bits to the non-reference B-frames. As a result, the PSNR distribution becomes more smooth and less hierarchical.

#### Table V

**BD-Rates of B-CANF as Compared Against Different Anchors Under Their Respective Settings**

| Anchor | I-frame codec | Intra-period GOP | BD-rate (%) | PSNR-RGB | BD-rate (%) | MS-SSIM-RGB |
|--------|---------------|------------------|-------------|-----------|-------------|-------------|
| ULTC [11] | BPG [55]/ICL 19 [56] | 10 | -53.7 | -55.4 | -50.3 | -56.9 | -44.2 | -48.5 | -38.7 | -44.8 |
| ULVC (ANFIC) | ANFIC [20]/ANFIC | 10 | -53.9 | -54.9 | -50.3 | -56.9 | -44.7 | -49.0 | -39.0 | -44.1 |
| RLV [22] | BPG/ICL'19 | 13 | -50.6 | -49.4 | -46.5 | -47.8 | -36.6 | -41.9 | -33.6 | -48.9 |
| RLV (ANFIC) | ANFIC/ANFIC | 13 | -47.8 | -46.9 | -44.1 | -46.9 | -34.0 | -39.3 | -35.3 | -46.1 |
| LHBDCC [13] | nubit2018-mean [47]/nubit2018-mean | 8 | -33.6 | -36.1 | -43.1 | -41.2 | -31.4 | - | - | - |
| LHBDCC (ANFIC) | ANFIC/ANFIC | 8 | -30.9 | -32.9 | -40.7 | -39.9 | - | - | - |
| B-EPIC [12] | NA/NA | infinity | -32.3 | -33.8 | -32.5 | - | 15.5 | -21.2 | -26.5 |

#### Table VI

**BD-Rate Comparison Under Different Intra-Periods:**

- **Intra-Period 32 (I32) and Intra-Period 8 (I8). The Anchor Is Our Proposed Method, B-CANF, Under the Respective Intra-Periods:**

| Anchor | BD-rate (%) | PSNR-RGB |
|--------|-------------|-----------|
| UVC | MCL-JCV | HEVC-B |
| I32 | I8 | I32 | I8 | I32 | I8 |
| B-CANF (ANFIC) | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Sheng'22 (ANFIC) | 14.6 | 20.4 | 9.2 | 13.6 | 14.0 | 22.1 |
| Li'22 (ANFIC) | -14.3 | 4.0 | -13.7 | -4.3 | -12.6 | 7.5 |

Fig. 6. Profiles of the per-frame PSNR’s and the per-frame bit rates averaged over GOPs on UVG dataset, with and without frame-type adaptive (FA) coding at high rates (1 λ = 2048). Frame 0, 32 are coded as I-frame, Frame 16 is coded as B*-frame, and the odd-numbered frames are non-reference B-frames while the even-numbered frames are reference B-frames.
Table VII

| Setting | Motion coding | Inter-frame coding | BD-rate (%) | PSNR-RGB |
|---------|---------------|--------------------|-------------|----------|
|         | Residual     | Conditional | Residual | Conditional | UVG | MCL-JCV | HEVC-B | CLIC’22 |
| A       | ✓            |            | ✓       |            | -47.5 | -41.1 | -46.9 | -33.8 |
| B       | ✓            | ✓          | ✓       |            | -41.8 | -36.3 | -42.2 | -30.0 |
| C       | ✓            |            | ✓       |            | -33.4 | -26.8 | -34.6 | -23.0 |
| D       | ✓            | ✓          | ✓       |            | -28.2 | -22.8 | -30.3 | -20.0 |

Table VIII

| Setting | Distortion weighting | FA module | BD-rate (%) | PSNR-RGB |
|---------|----------------------|-----------|-------------|----------|
|         |                      |           | UVG | MCL-JCV | HEVC-B | CLIC’22 |
| A       | ✓                    | ✓         | -47.5 | -41.1 | -46.9 | -33.8 |
| B       | ✓                    | ✓         | -41.4 | -36.3 | -42.2 | -30.0 |
| C       | ✓                    | ✓         | -33.4 | -26.8 | -34.6 | -23.0 |
| D       | ✓                    | ✓         | -28.2 | -22.8 | -30.3 | -20.0 |

Fig. 7. Comparison of PSNR-RGB and bit allocation by turning on and off frame-type adaptive coding in the motion and/or inter-frame codecs at high rates ($\lambda_1 = 2048$). The PSNR-RGB and bit-allocation results are averaged over B-frames of the test sequences in UVG dataset. FA refers to frame-type adaptive coding. Setting A is our B-CANF model.

Table IX further presents BD-rate results by turning on and off frame-type adaptive coding in the motion and/or inter-frame codecs. Interestingly, when applied to either the inter-frame codec (from D to C) or the motion codec (from D to B) alone, it is not as effective as the case when it is enabled for both codecs (from D to A).

To gain insights into this observation, Fig. 7 analyzes the bit allocation between motion coding and inter-frame coding for B-frames. Comparing setting C with setting D, we observe that enabling frame-type adaptive coding for the inter-frame codec only is able to reduce effectively the bit rate of inter-frame coding (green) while keeping the motion bit rate (red) relatively untouched. Likewise, the comparison of settings B and D shows that applying it to the motion codec only reduces effectively the motion bit rate (red). In this case, the slightly increased bit rate of inter-frame coding (green) may be attributed to the use of a shared inter-frame codec, which is unable to adapt to the changes in the inter-frame correlation that result from frame-type adaptive motion coding. Tuning on frame-type adaptive coding for both codecs allows the greatest degree of flexibility in allocating bits to different parts, achieving a synergy effect.

3) Coded Flow Maps for B*-frames: Figs. 8(b) and (c) visualize the two coded flow maps for B*-frames. As shown in Fig. 8(d), they are seen to disagree slightly with each other around object boundaries. Table X presents BD-rate results for three variants of motion compensation. Both refers to using two hypotheses, i.e., the coded flow map $\hat{m}_{t \rightarrow t-k}$ and its virtual counterpart after sign reversal, whereas First or Second refers to using solely one of them. We see that using solely $\hat{m}_{t \rightarrow t-k}$ (First) or its virtual counterpart (Second) for motion compensation leads to less rate saving than using both simultaneously. This may be attributed to the fact that the input flow map is usually less reliable in boundary regions. Performing multi-hypothesis prediction with slightly different motion estimates in these regions is an effective means of mitigating the reliability issue.

4) B*-frame vs. Separate P-frame Coding: Table XI investigates the effectiveness of B*-frame coding as a substitute for P-frame coding under various GOP sizes. The former reuses the B-frame codec while the latter needs to train end-to-end a separate, dedicated P-frame codec together with the other components of B-CANF. In most cases, our B*-frame...
mechanism is seen to achieve similar BD-rate results to the use of a separate P-frame codec. In particular, under GOP size 4, it performs even better than P-frame coding. Recall that B*-frames are essentially a multi-hypothesis prediction technique.

5) GOP Size vs. Rate-distortion Performance: We evaluate the effect of GOP size on the performance of B-CANF. A number of GOP sizes, including 4, 8, 16, are tested with intra-period 32. The BD-rates are summarized in Table XI (see the results w/o a separate P-frame codec). The corresponding rate-distortion curves on UVG dataset are presented in Fig. 9. From Fig. 9, the rate-distortion performance of B-CANF is seen to improve with the increased GOP size. The improvement is most obvious at low rates. Like P-frames, our B*-frames suffer more from temporal error propagation with smaller GOP sizes (in which cases, B*-frames are sent more frequently), especially at low rates where poor reconstruction and motion quality is expected. Increasing GOP size decreases the frequency of B*-frames, thereby reducing temporal error propagation.

6) The Number of Autoencoding Transforms: Table XII compares the BD-rates between 2-step and 1-step B-CANF to explore the effect of the number of autoencoding transforms on compression performance. The 1-step B-CANF is obtained by skipping the first autoencoding transform \( \{ A_1, S_1 \} \) in Fig. 1(b). In order to have a similar model size to the 2-step B-CANF, it has more channels in each autoencoding transform. Remarkably, the 1-step B-CANF reduces to a specific implementation of conditional VAE (CVAE). From Table XII, we observe that the gain of the 2-step B-CANF over the 1-step B-CANF is obvious across the datasets. The result justifies our use of CANF rather than CVAE.

E. Subjective Quality

Fig. 10 presents the subjective quality comparison between HM (random access), LHBDC [13] and our B-CANF. LHBDC (MSE) and B-CANF (MSE) are trained to optimize PSNR-RGB while B-CANF (SSIM) is our model trained with MS-SSIM-RGB. All the schemes are evaluated with GOP 16 and intra-period 32, and all the learning-based methods use ANFIC as the I-frame codec. It is seen that our B-CANF (MSE) achieves comparable or even better subjective quality than LHBDC (MSE), with its bit rate being nearly one order of magnitude smaller than that of LHBDC (MSE). Compared with HM (random access), our B-CANF (SSIM) preserves more texture details (cf. patterns on fingers in the first row, pillars in the second row, and textures in the last row) at a lower bit rate.

F. Complexity Analysis

Table XIII characterizes the complexity of the proposed B-CANF in terms of runtimes, MACs and model size. We also measure the memory footprint following [28]. This includes the size of the decoded picture buffer, which stores decoded

---

**TABLE IX**

| Setting | Frame-type adaptive coding | BD-rate (%) PSNR-RGB |
|---------|-----------------------------|----------------------|
|         | Motion codec | Inter-frame codec | UVG | MCL-JCV | HEVC-B | CLIC 22 |
| A       | ✓             | ✓                 | -47.5 | -41.1 | -46.9 | -33.8 |
| B       | ✓             | ✓                 | -41.6 | -32.7 | -40.6 | -28.4 |
| C       | ✓             | ✓                 | -42.3 | -35.9 | -43.2 | -30.9 |
| D       |               |                   | -39.4 | -32.1 | -41.0 | -28.7 |

**TABLE X**

| Hypothesis | BD-rate (%) PSNR-RGB |
|------------|----------------------|
|            | UVG | MCL-JCV | HEVC-B | CLIC 22 |
| Both       | -40.6 | -34.6 | -39.4 | -18.7 |
| First      | -35.0 | -31.0 | -35.1 | -14.3 |
| Second     | -35.2 | -30.9 | -35.1 | -14.7 |

**TABLE XI**

| GOP       | BD-rate (%) PSNR-RGB |
|-----------|----------------------|
|           | UVG | MCL-JCV | HEVC-B | CLIC 22 |
| w/o       | w/o | w/o | w/o | w/o | w/o |
| 16        | -47.5 | -46.4 | -41.1 | -46.9 | -47.7 | -33.8 | -36.5 |
| 8         | -45.7 | -44.8 | -39.5 | -39.3 | -43.1 | -46.3 | -29.7 | -30.8 |

**TABLE XII**

| Motion and Inter-frame codec | BD-rate (%) PSNR-RGB |
|------------------------------|----------------------|
|                              | UVG | MCL-JCV | HEVC-B | CLIC 22 |
| 1-step B-CANF (CVAE)         | -36.7 | -30.1 | -37.3 | -24.9 |
| 2-step B-CANF (Ours)         | -47.5 | -41.1 | -46.9 | -33.8 |
Fig. 10. Subjective quality comparison of HM (random access), LHBDC [13] and our B-CANF. The parentheses “(MSE)” and “(SSIM)” indicate that the model is trained with MSE and MS-SSIM, respectively. Zoom in for better visualization.

| Method       | Frame-type | Encode | Decode | Model Size | Decoded Picture Buffer | Peak Memory |
|--------------|------------|--------|--------|------------|-------------------------|-------------|
|              |            | Time   | Time   |            |                         |             |
|              |             |        |        |            |                         |             |
| HM [41]      | B          | 33.28s | 0.04s  | -          | -                       | -           |
| VTM [54]     | B          | 731.33s| -      | 0.07s      | -                       | -           |
| DCVC [29]    | P          | 7.70s  | 28.97s | 8M         | 3 full-res              | 128 full-res|
| CANF-VC [15] | P          | 1.45s  | 1.07s  | 31M        | 13 full-res             | 64 full-res |
| Sheng’22 [30]| P          | 0.82s  | 0.58s  | 10.7M      | 67 full-res             | 128 full-res|
| Li’22 [48]   | P          | 0.83s  | 0.61s  | 17.5M      | 67 full-res             | 128 full-res|
| LHBDC [13]   | B          | 1.19s  | 0.73s  | 23.5M      | 15 full-res             | 96 full-res |
| B-CANF (Ours)| B*         | 1.44s  | 1.06s  | 24M        | 3 full-res              | 64 full-res |
| B-CANF (Ours)| B          | 1.69s  | 1.09s  | 24M        | 15 full-res             | 64 full-res |

For frame-type P/B*, we use the prediction structure IPPP… under intra-period 32 and GOP size 32. The encoding/decoding times are then averaged over the first 100 P/B*-frames of Beauty sequence in UVG dataset. For frame-type B, the encoding/decoding times are averaged over the first 100 B-frames of the same sequence with intra-period 32, GOP size 16, and hierarchical B prediction.

The following observations can be made from Table XIII.

1. The prolonged runtimes of DCVC [29] are attributed...
to the use of an autoregressive model for entropy coding. In contrast, CANF-VC [15], LHBDC [13] and our B-CANF do not use any auto-regressive model for motion coding or inter-frame coding. (2) In comparison with VAE-based coding, DCVC [29] and LHBDC [13], the larger MAC of CANF-VC [15] and B-CANF comes from stacking multiple autoencoding transforms. Nevertheless, the comparable encoding/decoding runtimes suggest that these autoencoding transforms are parallel-friendly. (3) Our B-CANF has lower buffering requirements and peak memory when compared to other approaches. (4) With our B-CANF, the encoding time and MAC of B-frames are seen to be higher than those of B*-frames (the last two rows in Table XIII). This is because motion estimation is performed twice for B-frames, whereas it is carried out only once for B*-frames (just like P-frames). In contrast, the decoding times and MAC of both types of frame are quite close, even though B-frames incur extra computations in the motion prediction network (Fig. 3(a) vs. Fig. 3(b)). This indicates that the motion prediction network is not the major computation bottleneck during decoding. (5) As compared with CANF-VC [15], which supports only P-frame coding, the B-frame coding in B-CANF has higher encoding MAC because motion estimation is performed twice. In contrast, our B*-frames, similar to P-frames, have comparable encoding MAC to CANF-VC [15]. The decoding MAC’s of both B-frames and B*-frames are seen to be comparable to that of CANF-VC [15]. (6) In terms of model size, B-CANF is similar to LHBDC [13].

V. CONCLUSION

In this paper, we propose a CANF-based B-frame coding framework, known as B-CANF, that exploits the notion of conditional coding for both motion and inter-frame coding. It features B*-frames and frame-type adaptive coding. We show that (1) B*-frames allow greater flexibility in supporting various GOP sizes without the need for an extra P-frame codec, and (2) frame-type adaptive coding improves the bit allocation among B/B*-frames. Extensive experimental results confirm the superiority of B-CANF to the other state-of-the-art B-frame coding schemes. How to achieve even higher compression performance by addressing the domain shift issue and reducing the model’s complexity is our future work.

REFERENCES

[1] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao, “DVC: An end-to-end deep video compression framework,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 10998–11007.

[2] G. Lu, X. Zhang, W. Ouyang, L. Chen, Z. Gao, and D. Xu, “An end-to-end learning framework for video compression,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 10, pp. 3292–3308, Oct. 2021.

[3] 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 Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2020, pp. 8503–8512.

[4] J. Lin, D. Liu, H. Li, and F. Wu, “M-LVC: Multiple frames prediction for learned video compression,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 3534–3551.

[5] Z. Hu, Z. Chen, D. Xu, G. Lu, W. Ouyang, and S. Gu, “Improving deep video compression by resolution-adaptive flow coding,” in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2020, pp. 193–209.

[6] O. Rippel, A. G. Anderson, K. Tatwawadi, S. Nair, C. Lytle, and L. Bourdev, “ELF-VC: Efficient learned flexible-rate video coding,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2021, pp. 14459–14468.

[7] H. Liu et al., “Neural video coding using multiscale motion compensation and spatiotemporal context model,” IEEE Trans. Circuits Syst. Video Technol., vol. 31, no. 8, pp. 3182–3196, Aug. 2021.

[8] T. Ladune, P. Philippe, W. Hamidouche, L. Zhang, and O. Déforges, “Optical flow and mode selection for learning-based video coding,” in Proc. IEEE 22nd Int. Workshop Multimedia Signal Process. (MMSP), Sep. 2020, pp. 1–6.

[9] T. Ladune, P. Philippe, W. Hamidouche, L. Zhang, and O. Déforges, “Conditional coding for flexible learned video compression,” in Proc. Neural Comput., Inf. Theory Appl.-Workshop, 2021.

[10] A. Djelouah, J. Campos, S. Schaub-Meyer, and C. Schroers, “Neural inter-frame compression,” IEEE/CVF Int. Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 6627–6636.

[11] R. Pourreza and T. Cohen, “Extending neural P-frame codecs for B-frame coding,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Oct. 2021, pp. 6660–6669.

[12] M. A. Yilmaz and A. M. Tekalp, “End-to-end rate-distortion optimized learned hierarchical bi-directional video compression,” IEEE Trans. Image Process., vol. 31, pp. 974–983, 2022.

[13] C.-Y. Wu, N. Singhal, and P. Krahenbuhl, “Video compression through image interpolation,” in Proc. Eur. Conf. Comput. Vis. (ECCV), Sep. 2018, pp. 1–10.

[14] Y.-H. Ho, C.-P. Chang, P.-Y. Chen, A. Gnutti, and W.-H. Peng, “CANF-VC: Conditional augmented normalizing flows for video compression,” in Proc. Eur. Conf. Comput. Vis. (ECCV), Oct. 2022, pp. 207–223.

[15] Z. Chen, T. He, X. Jin, and F. Wu, “Learning for video compression,” IEEE Trans. Circuits Syst. Video Technol., vol. 30, no. 2, pp. 566–576, Feb. 2020.

[16] H. Liu, M. Lu, Z. Chen, X. Cao, Z. Ma, and Y. Wang, “End-to-end neural video coding using a compound spatiotemporal representation,” IEEE Trans. Circuits Syst. Video Technol., vol. 32, no. 8, pp. 5650–5662, Aug. 2022.

[17] M. Afonso, F. Zhang, and D. R. Bull, “Video compression based on spatio-temporal resolution adaptation,” IEEE Trans. Circuits Syst. Video Technol., vol. 29, no. 1, pp. 275–280, Jan. 2019.

[18] M.-J. Chen, H.-S. Xie, C. Chen, W.-D. Peng, and H.-M. Hang, “Learned hierarchical B-frame coding with adaptive feature modulation for YUV 4:2:0 content,” 2022, arXiv:2212.14187.

[19] Y.-H. Ho, C.-C. Chan, W.-H. Peng, H.-M. Hang, and M. Domanski, “ANFIC: Image compression using augmented normalizing flows,” IEEE Open J. Circuits Syst., vol. 2, pp. 613–626, 2021.

[20] Z. Hu, G. Lu, and D. Xu, “FVC: A new framework towards deep video compression in feature space,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 1502–1511.

[21] R. Yang, F. Mentzer, L. Van Gool, and R. Timofte, “Learning for video compression with recurrent auto-encoder and recurrent probability model,” IEEE J. Sel. Topics Signal Process., vol. 15, no. 2, pp. 388–401, Feb. 2021.

[22] A. Golinski, R. Pourreza, Y. Yang, G. Sautiere, and T. S. Cohen, “Extending neural P-frame codecs for B-frame coding,” 2022, arXiv:2207.14678.

[23] Z. Hu, G. Lu, J. Guo, S. Liu, W. Jiang, and D. Xu, “Coarse-to-fine deep video coding with hyperprior-guided mode prediction,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2022, pp. 5911–5920.

[24] A. Djelouah, J.-R. Ohm, W.-J. Han, and T. Wiegand, “Optical flow and mode selection for learning-based video coding,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. Pattern Recognit., Oct. 2021, pp. 1420–1429.

[25] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, “Overview of the Versatile Video Coding (VVC) standard,” IEEE Trans. Circuits Syst. Video Technol., vol. 32, no. 12, pp. 1649–1668, Dec. 2022.

[26] Z. Hu, G. Lu, Y. Wang, and T. Cohen, “Transformer-based transform coding,” in Proc. Int. Conf. Learn. Represent., 2022. [Online]. Available: https://openreview.net/forum?id=IDwN6jHkK8
[29] J. Li, B. Li, and Y. Lu, “Deep contextual video compression,” in Proc. Adv. Neural Inf. Process. Syst., 2021, pp. 1–13.

[30] X. Sheng, J. Li, B. Li, L. Li, D. Liu, and Y. Lu, “Temporal context mining for learned video compression,” IEEE Trans. Multimedia, early access, Nov. 8, 2022, doi: 10.1109/TMM.2022.3220421.

[31] F. Mentzer et al., “VCT: A video compression transformer,” 2022, arXiv:2206.07307.

[32] A. Vaswani et al., “Attention is all you need,” in Advances in Neural Information Processing Systems, vol. 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547de91fbad0523c14ca8f5aa-Paper.pdf

[33] T. Wiegnard, G. J. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the H.264/AVC video coding standard,” IEEE Trans. Circuits Syst. Video Technol., vol. 13, no. 7, pp. 560–576, Jul. 2003.

[34] C.-W. Huang, L. Dinh, and A. Courville, “Augmented normalizing flows: Bridging the gap between generative flows and latent variable models,” 2020, arXiv:2002.07101.

[35] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and T. Huang, “Free-form image inpainting with gated convolution,” in Proc. IEEE/CVF Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 4470–4479.

[36] X. Wang, K. Yu, C. Dong, and C. Change Loy, “Recovering realistic texture in image super-resolution by deep spatial feature transform,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 606–615.

[37] M. Song, J. Choi, and B. Han, “Variable-rate deep image compression through spatially-adaptive feature transform,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Los Alamitos, CA, USA, Oct. 2021, pp. 2360–2369, doi: 10.1109/ICCV48922.2021.00238.

[38] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. Huang, “Free-form image inpainting with gated convolution,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 4470–4479.

[39] [55] F. Bellard. (2015). BPG Image Format. [Online]. Available: https://bellard.org/bpg

[40] [56] E. P. S. J. Ballé and V. Laparra, “End-to-end optimized image compression,” in Proc. IEEE Int. Conf. Learn. Represent., 2020.

[41] [57] Z. Huang, T. Zhang, W. Heng, B. Shi, and S. Zhou, “Real-time intermediate flow estimation for video frame interpolation,” 2020, arXiv:2011.06294.

[42] [58] D. Fourure, R. Emont, E. Fromont, D. Muselet, A. Trembeau, and C. Wolf, “Residual conv-deconv grid network for semantic segmentation,” in Proc. Brit. Mach. Vis. Conf., 2017.

[43] [59] E. P. S. J. Ballé and V. Laparra, “End-to-end optimized image compression,” in Proc. Int. Conf. Learn. Represent., 2020.

[44] [60] [61] Z. Huang, T. Zhang, W. Heng, B. Shi, and S. Zhou, “Real-time intermediate flow estimation for video frame interpolation,” 2020, arXiv:2011.06294.

[45] [62] D. Fourure, R. Emont, E. Fromont, D. Muselet, A. Trembeau, and C. Wolf, “Residual conv-deconv grid network for semantic segmentation,” in Proc. Brit. Mach. Vis. Conf., 2017.

[46] [63] E. P. S. J. Ballé and V. Laparra, “End-to-end optimized image compression,” in Proc. Int. Conf. Learn. Represent., 2020.

[47] [64] Z. Huang, T. Zhang, W. Heng, B. Shi, and S. Zhou, “Real-time intermediate flow estimation for video frame interpolation,” 2020, arXiv:2011.06294.

[48] [65] D. Fourure, R. Emont, E. Fromont, D. Muselet, A. Trembeau, and C. Wolf, “Residual conv-deconv grid network for semantic segmentation,” in Proc. Brit. Mach. Vis. Conf., 2017.

[49] [66] E. P. S. J. Ballé and V. Laparra, “End-to-end optimized image compression,” in Proc. Int. Conf. Learn. Represent., 2020.

[50] [67] Z. Huang, T. Zhang, W. Heng, B. Shi, and S. Zhou, “Real-time intermediate flow estimation for video frame interpolation,” 2020, arXiv:2011.06294.

[51] [68] D. Fourure, R. Emont, E. Fromont, D. Muselet, A. Trembeau, and C. Wolf, “Residual conv-deconv grid network for semantic segmentation,” in Proc. Brit. Mach. Vis. Conf., 2017.

[52] [69] E. P. S. J. Ballé and V. Laparra, “End-to-end optimized image compression,” in Proc. Int. Conf. Learn. Represent., 2020.

[53] [70] Z. Huang, T. Zhang, W. Heng, B. Shi, and S. Zhou, “Real-time intermediate flow estimation for video frame interpolation,” 2020, arXiv:2011.06294.

[54] [71] D. Fourure, R. Emont, E. Fromont, D. Muselet, A. Trembeau, and C. Wolf, “Residual conv-deconv grid network for semantic segmentation,” in Proc. Brit. Mach. Vis. Conf., 2017.