CANF-VC: Conditional Augmented Normalizing Flows for Video Compression

Yung-Han Ho[1], Chih-Peng Chang[1], Peng-Yu Chen[1], Alessandro Gnutti[2], and Wen-Hsiao Peng[1]

[1] Department of Computer Science, National Yang Ming Chiao Tung University, Taiwan `wpeng@cs.nctu.edu.tw`
[2] Department of Information Engineering, CNIT - University of Brescia, Italy `alessandro.gnutti@unibs.it`

Abstract. This paper presents an end-to-end learning-based video compression system, termed CANF-VC, based on conditional augmented normalizing flows (CANF). Most learned video compression systems adopt the same hybrid-based coding architecture as the traditional codecs. Recent research on conditional coding has shown the sub-optimality of the hybrid-based coding and opens up opportunities for deep generative models to take a key role in creating new coding frameworks. CANF-VC represents a new attempt that leverages the conditional ANF to learn a video generative model for conditional inter-frame coding. We choose ANF because it is a special type of generative model, which includes variational autoencoder as a special case and is able to achieve better expressiveness. CANF-VC also extends the idea of conditional coding to motion coding, forming a purely conditional coding framework. Extensive experimental results on commonly used datasets confirm the superiority of CANF-VC to the state-of-the-art methods. The source code of CANF-VC is available at `https://github.com/NYCU-MAPL/CANF-VC`.

1 Introduction

Video compression is an active research area. The video traffic continues to grow exponentially due to an increased demand for various emerging video applications, particularly on social media platforms and mobile devices. The traditional video codecs, such as HEVC [33] and VVC [7], are still thriving towards being more efficient, hardware-friendly, and versatile. However, their backbones follow the hybrid-based coding framework–namely, spatial/temporal predictive coding plus transform-based residual coding–which has not changed since decades ago.

The arrival of deep learning spurs a new wave of developments in end-to-end learned image and video compression [30,9,28,15,23,32]. The seminal work [4] by Ballé *et al.* connects for the first time the learning of an image compression system to learning a variational generative model, known as the variational autoencoder (VAE) [19]. VAE involves learning the autoencoder network jointly with the prior distribution network by maximizing the variational lower bound (ELBO) on the image likelihood $p(x)$. Many follow-up works have been centered

around enhancing the autoencoder network [9,8] and/or improving the prior modeling [30,9]. Lately, there have been few attempts at introducing normalizing flow models [28,13] to learned image compression

Inspired by the success of learned image compression, research on learned video compression is catching up quickly. However, most end-to-end learned video compression systems [26,27,24,14,32] were developed based primarily on the traditional, hybrid-based coding architecture, replacing key components, such as inter-frame prediction and residual coding, with neural networks. The idea of residual coding is to encode a target frame x_t by coding the prediction residual $r_t = x_t - x_c$ between x_t and its motion-compensated reference frame x_c. The recent revisit of residual coding as a problem of conditional coding in [21,22,23] opens up a new dimension of thinking. Arguably, the entropy $H(x_t - x_c)$ of the residual between the coding frame x_t and its motion-compensated reference frame x_c is greater than or equal to the conditional entropy $H(x_t|x_c)$, i.e. $H(x_t - x_c) \geq H(x_t|x_c)$. How to learn $p(x_t|x_c)$ is apparently the key to the success of conditional coding.

In this paper, we present a conditional augmented normalizing flow-based video compression (CANF-VC) system, which is inspired partly by the ANF-based image compression (ANFIC) [13]. However, while ANFIC [13] adopts ANF to learn the (unconditional) image distribution $p(x)$ for image compression, we address video compression from the perspective of learning a video generative model by maximizing the conditional likelihood $p(x_t|x_c)$. We choose the conditional augmented normalizing flow (CANF) to learn $p(x_t|x_c)$, because ANF is a special type of generative model, which includes VAE as a special case and is able to achieve superior expressiveness to VAE.

Our work has three main contributions: (1) CANF-VC is the first normalizing flow-based video compression system that leverages CANF to learn a video generative model for conditional inter-frame coding; (2) CANF-VC extends the idea of conditional inter-frame coding to conditional motion coding, forming a purely conditional coding framework; and (3) extensive experimental results confirm the superiority of CANF-VC to the state-of-the-art methods.

2   Related Work

2.1   Learned Video Compression

End-to-end learned video compression is a hot research area. DVC [26] presents the first end-to-end learned video coding framework based on temporal predictive coding. Since then, there have been several improvements on learning-based motion-compensated prediction. Agustsson *et al.* [2] estimate the uncertainty about the flow map in forming a frame predictor, with a scale index sent for each pixel to determine a spatially-varying Gaussian kernel for blurring the reference frame. Liu *et al.* [25] perform feature-domain warping in a coarse-to-fine manner. Hu *et al.* [15] adopt deformable convolution for feature warping. Lin *et al.* [24] and Yang *et al.* [37] form a multi-hypothesis prediction from multiple reference frames. To reduce motion overhead, Lin *et al.* [24] use predictive motion coding by extrapolating a flow map predictor from the decoded flow maps.

Fig. 1: The architectures of ANF: (a) N-step ANF, (b) N-step hierarchical ANF, and (c) ANF for image compression (ANFIC).

Rippel *et al.* [32] use the flow map predictor for motion compensation and signal an incremental flow map between the resulting motion-compensated frame and the target frame. Hu *et al.* [14] adapt, either locally or globally, the resolution of the flow map features. Most learned video codecs encode the residual frame or the residual flow map by a variational autoencoder (VAE)-based image coder [4]. Some additionally leverage a recurrent neural network to propagate causal, temporal information in forming a temporal prior for entropy coding [38,25].

2.2   Conditional Coding

The idea of encoding the residual signal has recently been revisited from the information-theoretic perspective. Ladune *et al.* [21] show that coding a video frame x_t conditionally based on its motion-compensated reference frame x_c can achieve a lower entropy rate than coding the residual signal $x_t - x_c$ unconditionally. The fact motivates their converting the VAE-based residual coder into a conditional VAE by concatenating x_c and x_t for encoding, and their latent representations for decoding. The idea was extended in [22] for conditional motion coding, which encodes motion latents in an implicit, one-stage manner. However, Fabian *et al.* [6] show that these conditional VAE-based approaches [21,22] may suffer from the *bottleneck* issue; that is, the latent representation of x_c produced by a neural network for conditional decoding may not capture all the information of x_c, which serves as a condition for encoding x_t. Such information loss and asymmetry can harm the efficiency of conditional coding. Li *et al.* [23] improve the work in [21] by ensuring that the same information-rich latent representation of x_c is utilized for both conditional encoding and decoding. Likewise, the work in [12] creates the same coding context for conditional encoding and decoding via a feedback recurrent module that aggregates the past latent information. In common, these approaches do not evaluate any residual signal explicitly.

2.3   Augmented Normalizing Flows (ANF)

To learn properly the conditional distribution $p(x_t|x_c)$ for conditional coding, we turn to augmented normalizing flows (ANF), a special type of generative model

able to achieve superior expressiveness to VAE. Different from the vanilla flow models [20,18,10], ANF [16] augments the input x with an independent noise e_z (Fig. 1a), allowing the augmented noise to induce a complex marginal on x [16]. ANF contains the autoencoding transform g_π as a basic building block, where the encoding transformation g_π^{enc} from (x, e_z) to (x, z) and the decoding transformation g_π^{dec} from (x, z) to (y, z) are specified by

$$g_\pi^{enc}(x, e_z) = (x, s_\pi^{enc}(x) \odot e_z + m_\pi^{enc}(x)) = (x, z), \tag{1}$$

$$g_\pi^{dec}(x, z) = ((x - \mu_\pi^{dec}(z))/\sigma_\pi^{dec}(z), z) = (y, z), \tag{2}$$

respectively. s_π^{enc}, m_π^{enc}, μ_π^{dec} and σ_π^{dec} are element-wise affine transform parameters, and they are driven by neural networks parameterized by π.

The training of ANF aims to maximize the *augmented data likelihood*–namely, $\arg\max_\pi p_\pi(x, e_z) = p(g_\pi(x, e_z))|det(\partial g_\pi(x, e_z)/\partial(x, e_z))|$. Performing one autoencoding transformation $(y, z) = g_\pi(x, e_z)$ (known as the one-step ANF) is equivalent to training a VAE by maximizing the evidence lower bound (ELBO) on the log-marginal $\log p_\pi(x)$ [19]. As such, the learned image compression with the factorized prior [3] can be viewed as an one-step ANF that adopts a purely additive autoencoding transform (i.e. $s_\pi^{enc}(x) = \sigma_\pi^{dec}(z) = 1$) and an augmented noise $e_z \sim \mathcal{U}(-0.5, 0.5)$ modeling the uniform quantization. In this case, the latents y, z follow the standard Normal $\mathcal{N}(0, I)$ and the learned factorized prior, respectively. In particular, the hyperprior extension [4] has a similar structure to the *hierarchical ANF* (Fig. 1b), an enhanced form of ANF [16] with $g_\phi^{enc}, g_\phi^{dec}$ playing a similar role to the hyper codec. For better expressiveness, one can stack multiple one-step ANF's as the *multi-step ANF*. In [13], Ho *et al.* introduce the first ANF-based image compression (Fig 1c), which combines the multi-step and the hierarchical ANF's.

3   Proposed Method

3.1   Problem Statement

In this section, we formally define our task and objective. Let $x_{1:T} \in \mathbb{R}^{T \times 3 \times H \times W}$ denote a (RGB) video sequence of width W and height H to be encoded, and $\hat{x}_{1:T}$ the decoded video. The video compression task is to strike a good balance between the distortion $d(\hat{x}_{1:T}, x_{1:T})$ of the decoded video $\hat{x}_{1:T}$ and the rate $r(\hat{x}_{1:T})$ needed to represent it. When $T = 1$, the task reduces to image compression, of which the problem is cast as learning a VAE by maximizing the ELBO on the log-likelihood $\log p(x)$ in [4]. The same perspective is applicable to video compression yet with the aim of learning a VAE that maximizes the joint log-likelihood $\log p(x_{1:T})$. Because $p(x_{1:T})$ factorizes as $\prod_{t=1}^{T} p(x_t|x_{<t})$, with $x_{<t}$ representing collectively the video frames up to time instance $t - 1$, video compression is often done frame-by-frame by learning the conditional distribution $p(x_t|x_{<t})$. In our task, the decoded frames $\hat{x}_{<t}$ are used in place of $x_{<t}$.

With the traditional predictive coding framework, the ELBO on $\log p(x_t|\hat{x}_{<t})$ has a form of

$$E_{q(\hat{f}_t, \hat{r}_t|x_t, \hat{x}_{<t})} \log p(x_t|\hat{f}_t, \hat{r}_t, \hat{x}_{<t}) - D_{KL}(q(\hat{f}_t, \hat{r}_t|x_t, \hat{x}_{<t})||p(\hat{f}_t, \hat{r}_t|\hat{x}_{<t})), \tag{3}$$

where the latents $\hat{f}_t \in \mathbb{R}^{2 \times H \times W}, \hat{r}_t \in \mathbb{R}^{3 \times H \times W}$ represent the (quantized) optical flow map and the (quantized) motion-compensated residual frame associated with $x_t \in R^{3 \times H \times W}$, respectively. The encoding distribution $q(\hat{f}_t, \hat{r}_t | x_t, \hat{x}_{<t}) = q(\hat{f}_t | x_t, \hat{x}_{<t}) q(\hat{r}_t | \hat{f}_t, x_t, \hat{x}_{<t})$ specifies the generation of $\hat{f}_t, \hat{r}_t$, while the decoding distribution $p(x_t | \hat{f}_t, \hat{r}_t, \hat{x}_{<t}) = \mathcal{N}(\hat{r}_t + warp(\hat{x}_{t-1}; \hat{f}_t), \frac{1}{2\lambda} I)$ models the reconstruction process of x_t, with $warp(\hat{x}_{t-1}; \hat{f}_t)$ denoting the backward warping of $\hat{x}_{t-1}$ based on $\hat{f}_t$, and $\frac{1}{2\lambda}$ being the variance of the Gaussian. Assuming the use of uniform quantization function for obtaining $\hat{f}_t, \hat{r}_t$, the Kullback–Leibler (KL) divergence $D_{KL}(\cdot || \cdot)$ evaluates to the rate costs associated with their transmission:

$$D_{KL}(q(\hat{f}_t, \hat{r}_t | x_t, \hat{x}_{<t}) || p(\hat{f}_t, \hat{r}_t | \hat{x}_{<t})) \tag{4}$$
$$= E_{q(\hat{f}_t, \hat{r}_t | x_t, \hat{x}_{<t})}(- \log p(\hat{f}_t | \hat{x}_{<t}) - \log p(\hat{r}_t | \hat{f}_t, \hat{x}_{<t})).$$

Substituting Eq. (4) into Eq. (3) and applying the law of total expectation yields

$$E_{q(\hat{f}_t | x_t, \hat{x}_{<t})}(RD_r(x_t | \hat{f}_t, \hat{x}_{<t}) + \log p(\hat{f}_t | \hat{x}_{<t})), \tag{5}$$

where

$$RD_r(x_t | \hat{f}_t, \hat{x}_{<t}) = E_{q(\hat{r}_t | \hat{f}_t, x_t, \hat{x}_{<t})}(\log p(x_t | \hat{f}_t, \hat{r}_t, \hat{x}_{<t}) + \log p(\hat{r}_t | \hat{f}_t, \hat{x}_{<t})), \tag{6}$$

which bears the interpretation of the ELBO on $\log p(x_t | \hat{f}_t, \hat{x}_{<t})$, with the latent being the quantized residual frame $\hat{r}_t$.

From Eqs. (5) and (6), we see that the traditional predictive coding of a video frame x_t includes (1) encoding the residual frame $\hat{r}_t$ based on $\hat{f}_t, \hat{x}_{<t}$ in order to maximize the log-likelihood $\log p(x_t | \hat{f}_t, \hat{x}_{<t})$ and (2) encoding the flow map $\hat{f}_t$ in a way that strikes a good balance between the maximization of $\log p(x_t | \hat{f}_t, \hat{x}_{<t})$ and the (negative) rate $E_{q(\hat{f}_t | x_t, \hat{x}_{<t})} \log p(\hat{f}_t | \hat{x}_{x<t})$ needed to signal $\hat{f}_t$.

In this work, we propose to turn the maximization of the log-likelihood $\log p(x_t | \hat{f}_t, \hat{x}_{<t})$, i.e. Eq. (6), into a problem of conditional coding, where $\hat{f}_t, \hat{x}_{<t}$ are utilized to formulate the motion-compensated frame $x_c \in \mathbb{R}^{3 \times H \times W}$ as a condition. Unlike the existing works [21,22,23,12], which adopt the conditional VAE, our conditional coder is constructed based on multi-step CANF in modeling $p(x_t | x_c)$ for its better expressiveness.

3.2   System Overview

Fig. 2a depicts our CANF-based video compression system, abbreviated as CANF-VC. It includes two major components: (1) the CANF-based inter-frame coder $\{G_\pi, G_\pi^{-1}\}$ and (2) the CANF-based motion coder $\{F_\pi, F_\pi^{-1}\}$. The inter-frame coder encodes a video frame x_t conditionally, given the motion-compensated frame x_c. It departs from the conventional residual coding by maximizing the conditional log-likelihood $p(x_t | x_c)$ with CANF model (Section 3.3). The motion
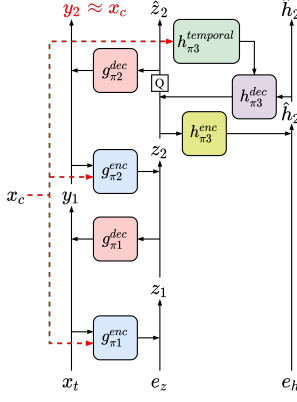
Fig. 2: Illustration of (a) the proposed CANF-VC framework and (b) the CANF-based inter-frame coder $\{G_\pi, G_\pi^{-1}\}$. The CANF-based motion coder $\{F_\pi, F_\pi^{-1}\}$ follows the same design as the inter-frame coder, with x_t, x_c replaced by f_t, f_c, respectively.

coder shares a similar architecture to the inter-frame coder. It extends conditional coding to motion coding, in order to signal the flow map f_t, which characterizes the motion between x_t and its reference frame $\hat{x}_{t-1}$. In our work, f_t is estimated by PWC-Net [34]. The compressed flow map $\hat{f}_t$ serves to warp the reference frame $\hat{x}_{t-1}$, with the warped result enhanced further by a motion compensation network to arrive at x_c. To formulate a condition for conditional motion coding, we introduce a flow extrapolation network to extrapolate a flow map f_c from three previously decoded frames $\hat{x}_{t-1}, \hat{x}_{t-2}, \hat{x}_{t-3}$ and two decoded flow maps $\hat{f}_{t-1}, \hat{f}_{t-2}$. Note that we expand the condition of $p(x_t|\hat{x}_{<t})$ from previously decoded frames $\{\hat{x}_{<t}\}$ to include also previously decoded flows $\{\hat{f}_{<t}\}$.

3.3   CANF-based Inter-frame Coder

Fig. 2b presents the architecture of our CANF-based inter-frame coder, which aims to learn the conditional distribution $p(x_t|x_c)$ of the coding frame x_t given the motion-compensated frame x_c. This is achieved by maximizing the augmented likelihood $p(x_t, e_z, e_h|x_c)$ in the CANF framework, where $e_z \in \mathbb{R}^{C \times \frac{H}{16} \times \frac{W}{16}}$, $e_h \in \mathbb{R}^{C \times \frac{H}{64} \times \frac{W}{64}}$ are the two augmented noise inputs. It is shown in [16] that maximizing $p(x_t, e_z, e_h|x_c)$ is equivalent to maximizing a lower bound on the marginal likelihood $p(x_t|x_c)$.

Architecture: Motivated by [13], our conditional inter-frame coder is a hybrid of the two-step and the hierarchical ANF's. The two autoencoding transforms $\{g_{\pi_1}^{enc}, g_{\pi_1}^{dec}\}, \{g_{\pi_2}^{enc}, g_{\pi_2}^{dec}\}$ convert x_t, e_z into their latents y_2, z_2, respectively, while the hierarchical autoencoding transform $\{h_{\pi_3}^{enc}, h_{\pi_3}^{dec}\}$ acts as the hyperprior codec, encoding the latent z_2 into the hyperprior representation $\hat{h}_2$. The volume preserving property of CANF requires that the latents y_2, z_2 (or $\hat{z}_2$), $\hat{h}_2$ have the

same dimensions as their respective inputs x_t, e_z, e_h. One notable distinction between CANF and ANFIC [13] is the incorporation of the condition x_c into the autoencoding transforms and the prior distribution, as will be detailed next.

Conditional Encoding: The core idea of our conditional coding is to let the latent y_2, which represents a transformed version of the target frame x_t, approximate the condition x_c, with the latents $z_2, \hat{h}_2$ encoding the information necessary for instructing the transformation. Specifically, the two autoencoding transforms operate similarly and successively. Taking $\{g_{\pi_1}^{enc}, g_{\pi_1}^{dec}\}$ as an example (see Fig.2b), we have

$$g_{\pi_1}^{enc}(x_t, e_z|x_c) = (x_t, e_z + m_{\pi_1}^{enc}(x_t, x_c)) = (x_t, z_1), \tag{7}$$

$$g_{\pi_1}^{dec}(x_t, z_1) = (x_t - \mu_{\pi_1}^{dec}(z_1), z_1) = (y_1, z_1). \tag{8}$$

That is, the one-step transformation from x_t to y_1 is done by subtracting the decoder output $\mu_{\pi_1}^{dec}(z_1)$ from x_t. Note that $\mu_{\pi_1}^{dec}(z_1)$ decodes the latent z_1, which aggregates the information of x_t, x_c, and the augmented noise e_z. We remark that the encoding process is made conditional on x_c by concatenating x_c and x_t to form the encoder input. Intuitively, supplying x_c as an auxiliary signal should ease the transformation from x_t to x_c. This process is repeated by taking y_1 and z_1 as the inputs to the next autoencoding transform $\{g_{\pi_2}^{enc}, g_{\pi_2}^{dec}\}$. In fact, the number of the autoencoding transforms is flexible. In comparison with Eqs. (1) and (2), our autoencoding transform is purely additive (i.e. $s_\pi^{enc}, \sigma_\pi^{dec}$ in Eqs. (1) and (2) are set to 1), which is found beneficial in terms of training stability.

The hierarchical autoencoding transform $\{h_{\pi_3}^{enc}, h_{\pi_3}^{dec}\}$ serves to estimate the probability distribution of z_2 for entropy coding. It operates according to

$$h_{\pi_3}^{enc}(z_2, e_h) = (z_2, e_h + m_{\pi_3}^{enc}(z_2)) = (z_2, \hat{h}_2), \tag{9}$$

$$h_{\pi_3}^{dec}(z_2, \hat{h}_2|x_c) = (\lfloor z_2 - \mu_{\pi_3}^{dec}(\hat{h}_2, h_{\pi_3}^{temporal}(x_c)) \rceil, \hat{h}_2) = (\hat{z}_2, \hat{h}_2), \tag{10}$$

where $\lfloor \cdot \rceil$ (depicted as Q in Fig. 2b) denotes the nearest-integer rounding, which is needed to express z_2 in fixed-point representation for lossy compression. At training time, the rounding effect is modeled by additive quantization noise. It is worth noting that x_c is provided as an auxiliary input to $\mu_{\pi_3}^{dec}$ to exert a combined effect of the hyperprior and the temporal prior ($h_{\pi_3}^{temporal}$ in Fig. 2b).

Conditional Decoding: The decoding process of our inter-frame coder updates the motion-compensated frame x_c successively to reconstruct x_t. It starts by entropy decoding the latents $\hat{z}_2, \hat{h}_2$, and substituting x_c for y_2. The quantized z_2 will then be recovered and decoded to reconstruct $\mu_{\pi_2}^{dec}(z_2)$, which updates x_c as $y_1 = x_c + \mu_{\pi_2}^{dec}(z_2)$. Subsequently, y_1 will be encoded conditionally based on x_c using $m_{\pi_2}^{enc}(y_1, x_c)$ in order to update the latent z_2 as $z_1 = z_2 - m_{\pi_2}^{enc}(y_1, x_c)$. Finally, z_1 is decoded by $\mu_{\pi_1}^{dec}(z_1)$ to update y_1 as the reconstructed version $\hat{x}_t = y_1 + \mu_{\pi_1}^{dec}(z_1)$ of x_t. In a sense, the reconstruction of x_t is achieved by passing the latent $\hat{z}_2$ through the composition of the decoding and encoding transforms to update x_c.

Conditional Prior Distribution: Another strategy we adopt to learn $p(x_t, e_z, e_h|x_c)$ is to introduce a conditional prior distribution $p(y_2, \hat{z}_2, \hat{h}_2|x_c)$.

Specifically, we assume that it factorizes as follows:

$$p(y_2, \hat{z}_2, \hat{h}_2 | x_c) = p(y_2|x_c)p(\hat{z}_2|\hat{h}_2, x_c)p(\hat{h}_2). \tag{11}$$

Because we require y_2 to approximate x_c, it is natural to choose $p(y_2|x_c)$ to be $\mathcal{N}(x_c, \frac{1}{2\lambda_1}I)$ with a small variance $\frac{1}{2\lambda_1}$. Moreover, following the hyperprior [4], $p(\hat{z}_2|\hat{h}_2, x_c)$ and $p(\hat{h}_2)$ are modeled by

$$p(\hat{z}_2|\hat{h}_2, x_c) = \mathcal{N}(0, (\sigma_{\pi_3}^{dec}(\hat{h}_2, h_{\pi_3}^{temporal}(x_c)))^2 I) * \mathcal{U}(-0.5, 0.5)$$
$$p(\hat{h}_2) = \mathcal{P}_{\hat{h}_2|\psi} * \mathcal{U}(-0.5, 0.5),$$

where $*$ denotes convolution and $P_{\hat{h}_2|\psi}$ is a factorized prior parameterized by ψ. The use of the motion-compensated frame x_c along with $\hat{h}_2$ in estimating the distribution of $\hat{z}_2$ combines temporal prior ($h_{\pi_3}^{temporal}$ in Fig. 2b) and hyperprior.

   Augmented Noises e_z, e_h**:** In the theory of ANF [16], the augmented noises are meant to induce a complex marginal on the input x. For the compression task, we fix e_z at 0 during training and test, in order not to increase the entropy rate at $\hat{z}_2$. For training, the quantization Q in Fig. 2b is simulated by additive noise. In contrast, we draw $e_h \sim \mathcal{U}(-0.5, 0.5)$ for simulating the quantization of the hyperprior at training time, and set it to zero at test time when the hyperprior is actually rounded.

   Extension to Conditional Motion Coding: The CANF-based motion coder follows the same design as the CANF-based inter-frame coder. The coding frame x_t is replaced with the optical flow map f_t and the motion-compensated frame x_c with the extrapolated flow map f_c. In addition, the temporal prior takes the extrapolated frame $warp(\hat{x}_{t-1}; f_c)$ as input. To perform the flow map extrapolation, we adopt a U-Net-based network (see supplementary document).

   In the supplementary document, we provide another CANF implementation, which additionally accepts x_c as input to the decoding transforms. We choose the current implementation due to its comparable performance and simpler design.

3.4   Training Objective

We train the conditional inter-frame and motion coders end-to-end. Inspired by Eq. (5), we first turn the maximization of the ELBO (i.e. RD_r in Eq. (6)) on $\log p(x_t|\hat{f}_t, \hat{x}_{<t})$ into maximizing $\log p(x_t, e_z, e_h|x_c)$. That is, to minimize

$$-\log p(x_t, e_z, e_h|x_c) = -\log p(\hat{h}_2) - \log p(\hat{z}_2|\hat{h}_2, x_c)$$
$$+\lambda_1 \|y_2 - x_c\|^2 - log \left| det \frac{\partial G_\pi(x_t, e_z, e_h|x_c)}{\partial(x_t, e_z, e_h)} \right|.$$

To ensure the reconstruction quality, we follow [13] to replace the negative log-determinant of the Jacobian with a weighted reconstruction loss $\lambda_2 d(x_t, \hat{x}_t)$, arriving at

$$-\log p(x_t, e_z, e_h|x_c) \approx \underbrace{-\log p(\hat{h}_2) - \log p(\hat{z}_2|\hat{h}_2, x_c)}_{R} + \lambda_1 \|y_2 - x_c\|^2 + \underbrace{\lambda_2 d(x_t, \hat{x}_t)}_{D},$$

which includes the rate R needed to signal the transformation between x_t and x_c, the regularization term requiring y_2 to approximate x_c, and the distortion D of $\hat{x}_t$. To complete the loss function, we also follow the second term in Eq. (5) to include the conditional motion rate used to signal f_t given f_c, which leads to

$$
\begin{aligned}
\mathcal{L} = &- \log p(\hat{h}_2) - \log p(\hat{z}_2 | \hat{h}_2, x_c) + \lambda_1 \|y_2 - x_c\|^2 \\
&- \log p_f(\hat{h}_2) - \log p_f(\hat{z}_2 | \hat{h}_2, f_c) + \lambda_2 d(x_t, \hat{x}_t),
\end{aligned}
\tag{12}
$$

where p_f (respectively, p) describes the prior distribution over the motion (respectively, inter-frame) latents.

3.5   Comparison with ANFIC and Other VAE-based Schemes

Our CANF-VC is based on ANF, as well as ANFIC, a learned image compression system proposed in [13]. However, they significantly differ from each other, not only because they refer to different applications. ANFIC [13] adopts an *unconditional* ANF to learn the image distribution $p(x)$ for image compression, whereas CANF-VC uses two *conditional* ANF's (CANF's) to learn the conditional distribution $p(x_t | x_c)$ for inter-frame coding and the conditional rate needed to signal the motion part, respectively. As a result, CANF-VC is a complete video coding framework. Note that how the conditional information x_c and f_c are both incorporated in the respective autoencoding transforms and in the respective prior distributions is first proposed in this work.

   CANF-VC is also distinct from conditional VAE-based frameworks, such as DCVC [23] and [22]. CANF-VC bases the compression backbone on CANF, which is a flow-based model and includes VAE as a special case. As compared with DCVC [23], CANF-VC additionally features conditional motion coding. Although conditional motion coding also appears in [22], their VAE-based approach does not explicitly estimate a flow map prior to conditional coding, and may suffer from the bottleneck issue [6] (Section 2.2). In contrast, CANF-VC takes an explicit approach and avoids the bottleneck issue by using the same x_c symmetrically in the encoder and the decoder due to its invertible property.

4   Experiments

4.1   Settings and Implementation Details

Training Details: We train our model on Vimeo-90k [36] dataset, which contains 91,701 7-frame sequences with resolution 448×256. We randomly crop these video clips into 256×256 for training. We adopt the Adam [17] optimizer with the learning rate 10^{-4} and the batch size 32. Separate models are trained to optimize first the mean-square error with $\lambda_2 = \{256, 512, 1024, 2048\}$ and $\lambda_1 = 0.01 * \lambda_2$ (see Eq. (12)). We then fine-tune these models for Multi-scale Structural Similarity Index (MS-SSIM), with λ_2 set to $\{4, 8, 16, 32, 64\}$. All the low-rate models are adapted from the one trained for the highest rate point.

   Evaluation Methodologies: We evaluate our models on commonly used datasets, including UVG [29], MCL-JCV [35], and HEVC Class B [11]. We follow

	Intra coder	BD-rate (%) PSNR-RGB			BD-rate (%) MS-SSIM-RGB			Size
	(PSNR-RGB/MS-SSIM-RGB)	UVG	MCL-JCV	HEVC-B	UVG	MCL-JCV	HEVC-B	
DVC_Pro [27]	-/-	-3.0	-	-13.1	-5.2	-	-20.8	29M
M-LVC [24]	BPG/BPG	-15.3	18.8	-38.6	-0.2	4.7	-37.9	-
RaFC [14]	hyperprior/hyperprior	-11.1	4.4	-9.3	-25.5	-27.9	-37.2	-
FVC [15]	BPG/BPG	-16.9	-3.8	-17.8	-45.0	-46.1	-54.3	26M
HM (LDP, 4 refs)	-/-	-29.4	-13.9	-29.6	-18.9	-13.8	-17.1	-
CANF-VC*	BPG/BPG	-35.5	**-14.6**	-35.4	-46.6	**-46.7**	-53.2	31M
DCVC [23]	cheng2020-anchor/hyperprior	-23.8	-14.4	-34.9	-43.9	-44.9	-50.7	8M
DCVC (ANFIC)	ANFIC/ANFIC	-24.8	-13.6	-34.0	-41.9	-43.7	-51.1	8M
CANF-VC Lite	ANFIC/ANFIC	**-37.3**	-14.3	**-39.8**	**-47.6**	-44.2	**-56.8**	15M
CANF-VC	ANFIC/ANFIC	**-42.5**	**-21.0**	**-40.1**	**-51.4**	**-47.6**	**-54.7**	31M

Table 1: BD-rate comparison with GOP size 10/12. The anchor is x265 in veryslow mode. The best performer is marked in **red** and the second best in **blue**.

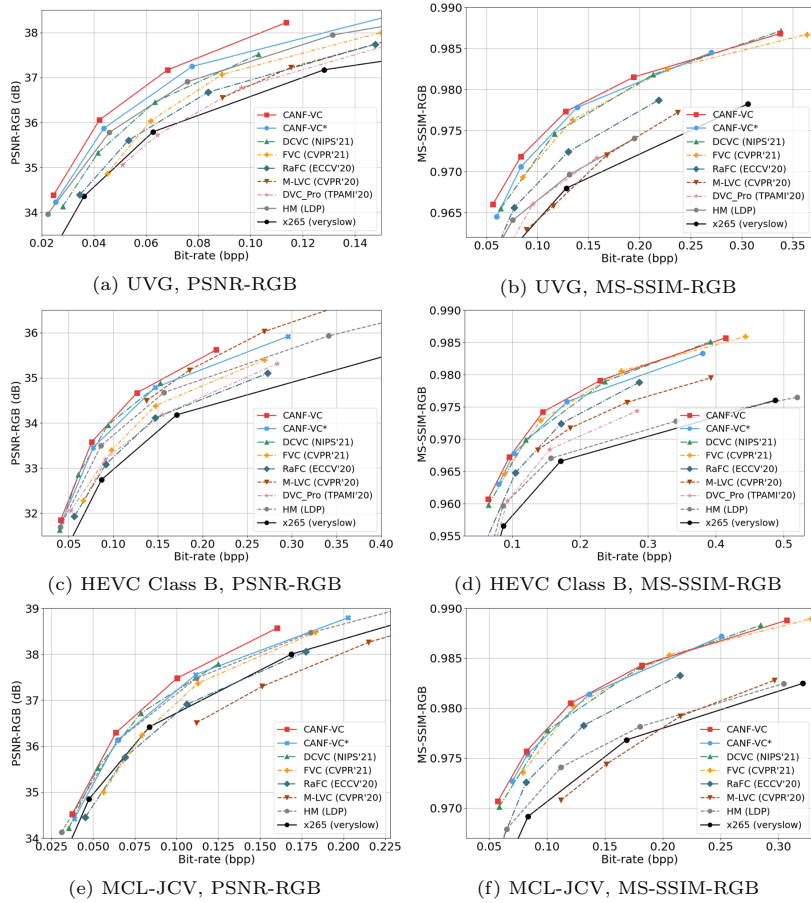common test protocols to provide results in Table 1 for 100-frame encoding with GOP [1] size 10 on HEVC Class B, and full-sequence encoding with GOP size 12 on the other datasets. Additionally, we present results for GOP size 32 in Table 2, to underline the contributions of our inter-frame and motion coders. For this additional setting, all the learned codecs use ANFIC [13] as the intra-frame coder and encode only the first 96 frames in every test sequence. To evaluate the rate-distortion performance, the bit rates are measured in bits per pixel (bpp), and the quality in PSNR-RGB and MS-SSIM-RGB. Moreover, we use x265 in veryslow mode as the anchor for reporting BD-rates.

Baseline Methods: The baseline methods for comparison include x265, HEVC Test Model (HM) [1] and several recent publications, including DVC_Pro [27], M-LVC [24], RaFC [14], FVC [15] and DCVC [23]. Because these baseline methods adopt different intra-frame coders (see the second column of Table 1), which are critical to the overall rate-distortion performance, we provide results with ANFIC [13] (CANF-VC) and BPG (CANF-VC*) as the intra-frame coders to ease comparison. Note that ANFIC [13] shows comparable performance to cheng2020-anchor [5]. It is to be noted that x265, HM [1], DVC_Pro [27], and M-LVC [24] use the same model optimized for PSNR to report PSNR-RGB and MS-SSIM-RGB results. While the other methods train separate models in reporting these results. We also present CANF-VC$^-$ and CANF-VC Lite as two additional variants of CANF-VC. CANF-VC$^-$ disables conditional motion coding while CANF-VC Lite implements a lightweight version of CANF-VC by reducing the channels in the autoencoding and the hyperprior transforms, and adopting SPyNet [31] as the flow estimation network.

4.2   Rate-Distortion and Subjective Quality Comparison

Rate-Distortion Comparison: The upper part of Table 1 compares the competing methods with their intra-frame coders, e.g. hyperprior [3], performing comparably to BPG. We see that our CANF-VC* (with BPG as the intra-frame coder) outperforms most of these baselines across different datasets in terms of PSNR-RGB. Its slight rate inflation (3%) as compared to M-LVC [24] on

---

[1] GOP refers to Group-of-Pictures and is often used interchangeably with the intra period in papers on learned video codecs.

	Intra coder	BD-rate (%) PSNR-RGB			BD-rate (%) MS-SSIM-RGB		
	(PSNR-RGB/MS-SSIM-RGB)	UVG	MCL-JCV	HEVC-B	UVG	MCL-JCV	HEVC-B
M-LVC (ANFIC)	ANFIC/ANFIC	-12.1	-5.3	-9.7	-7.5	-8.4	-18.8
DCVC (ANFIC)	ANFIC/ANFIC	-16.3	-21.3	-10.5	**-38.8**	**-48.9**	-39.3
CANF-VC Lite	ANFIC/ANFIC	**-36.1**	-26.5	**-30.3**	-37.9	-47.8	**-44.2**
CANF-VC$^-$	ANFIC/ANFIC	-31.1	-29.5	-23.6	-36.2	-47.8	-38.0
CANF-VC	ANFIC/ANFIC	-35.9	**-32.0**	-27.7	**-40.3**	**-49.6**	**-41.3**
HM (LDP, 4 refs)	-/-	**-41.6**	**-38.6**	**-32.1**	-34.3	-32.0	-31.0

Table 2: BD-rate comparison with GOP size 32. All the competing methods (except HM) use ANFIC [13] as the intra-frame coder. The anchor is x265 in veryslow mode. The best performer is marked in **red** and the second best in **blue**.

Fig. 3: Rate-distortion performance evaluation with GOP size 10/12 on UVG, HEVC Class B, and MCL-JCV datasets for both PSNR-RGB and MS-SSIM-RGB.

HEVC-B class may be attributed to the not-fully-aligned rate range in which the BD-rate is measured (see Fig. 3). Note that M-LVC [24] is initially trained for GOP size 100. With no access to its training software, a rate shift occurs

Ground Truth	DCVC (ANFIC)	CANF-VC	DCVC-ssim (ANFIC)	CANF-VC-ssim
	PSNR-RGB: 27.71 dB 0.0441 bpp	PSNR-RGB: 29.00 dB 0.0396 bpp	MS-SSIM-RGB: 0.952 0.0425 bpp	MS-SSIM-RGB: 0.955 0.0465 bpp
	PSNR-RGB: 29.29 dB 0.0373 bpp	PSNR-RGB: 30.23 dB 0.0294 bpp	MS-SSIM-RGB: 0.946 0.0580 bpp	MS-SSIM-RGB: 0.947 0.0573 bpp

Fig. 4: Subjective quality comparison between CANF-VC and DCVC (ANFIC).

when its test code is re-run for GOP size 10/12. Another observation is that CANF-VC* shows similar MS-SSIM-RGB results to FVC [15], while surpassing the others considerably. The lower part of Table 1 further shows that in terms of both quality metrics, our full model CANF-VC performs consistently better than both DCVC variants, where one uses ANFIC [13] and the other adopts cheng2020-anchor [5] as their respective intra-frame coders. The same observation can be made with CANF-VC$^-$ and CANF-VC Lite, except that they perform similarly to DCVC [23] on MCL-JCV.

Under the long GOP setting (Table 2), the gain of our schemes (all three variants) over DCVC (ANFIC) and M-LVC (ANFIC) becomes more significant in terms of PSNR-RGB, while CANF-VC$^-$ and CANF-VC Lite show comparable or better MS-SSIM-RGB results than DCVC (ANFIC). Interestingly, the gap in PSNR-RGB between the more capable HM and the learned coders is still considerable, although the latter outperform HM in terms of MS-SSIM-RGB.

Subjective Quality Comparison: Fig. 4 presents a subjective comparison between our CANF-VC and DCVC (ANFIC). Both schemes are trained for PSNR-RGB and MS-SSIM-RGB, use ANFIC as the intra-frame coder, and set GOP size to 32. Our CANF-VC is seen to preserve better the shape of the objects and has no color bias, as compared to DCVC (ANFIC).

4.3   Ablation Experiments

In this section, unless otherwise stated, all the experiments are conducted on UVG dataset [29], with the BD-rates reported against x265 in veryslow mode.

Conditional Inter-frame Coding vs. Residual Coding: To single out the gain of conditional inter-frame coding over residual coding, Table 3a presents a breakdown analysis in terms of BD-rate savings. In this ablation experiment, the conditional motion coding is disabled and replaced with the motion coder

from DVC [26]. Besides, the residual coding schemes adopt ANFIC [13] for coding the residual frame $x_t - x_c$ as an intra image. The variants with the temporal prior additionally involve the motion-compensated frame x_c in estimating the coding probabilities of the latent code (i.e. $\hat{z}_2$ in Fig. 2b). As seen in the table, the conditional inter-frame coding outperforms the residual coding significantly, whether the temporal prior is enabled or not. This suggests that a direct application of ANFIC to residual coding is unable to achieve the same level of gain as our CANF-based inter-frame coding. The temporal prior additionally improves the rate savings of both schemes by 2.5% to 4.2%.

Conditional Motion Coding vs. Predictive/Intra Motion Coding: This ablation experiment addresses the benefits of conditional motion coding. To this end, different competing settings employ the same conditional inter-frame coding, but change the way the flow map f_t is coded. The baseline settings use ANFIC [13] to code f_t as an intra image or the flow map residual $f_t - f_c$ without any condition. For the conditional motion coding, we additionally present results by simply using the previously decoded flow map $\hat{f}_{t-1}$ as the condition. Separate models are trained for each test case. From Table 3b, our conditional motion coding (i.e. coding f_t based on f_c) achieves the best performance. In terms of rate savings, its gain over the two unconditional variants, i.e. coding f_t or $f_t - f_c$ unconditionally, is quite significant. This result corroborates the superiority of our conditional motion coding to predictive motion coding (i.e. coding $f_t - f_c$). As expected, the quality of the condition has a crucial effect on compression performance. The trivial use of the previously decoded flow $\hat{f}_{t-1}$ does not show much gain as compared to unconditional coding. The fact substantiates the effectiveness of our extrapolation network.

The Number of Autoencoding Transforms: Table 3c explores the effect of the number of autoencoding transforms on compression performance. The 1-step models are obtained by skipping the autoencoding transform $\{g_{\pi_1}^{enc}, g_{\pi_1}^{dec}\}$ in Fig. 2b. To have the model size compatible with the 2-step models, the 1-step models have more channels in every autoencoding transform. We first experiment with the conditional inter-frame coding, with the motion coder from DVC [26]. In this case, the 2-step model improves the rate saving of the 1-step model by 1.7%. Given the 2-step inter-frame coder, it is further seen that the 2-step motion coder also improves the rate saving of the 1-step motion coder by 4.4%. This suggests that with a similar model size, the 2-step model is superior to the 1-step model in both inter-frame and motion coding.

Table 3d complements Table 3c to present results for 1-, 2- and 3-step CANF when applied to both the motion and inter-frame codecs. 3-step CANF extends straightforwardly the 2-step CANF by incorporating one additional autoencoding transform. Despite a larger capacity, the 3-step CANF performs worse than the 2-step CANF and comparably to the 1-step CANF. From Fig. 2b, the quantization error introduced to the latent code z_2 and the approximation error between x_c (used for decoding) and y_2 (generated during encoding) are propagated and accumulated (from top to bottom in Fig. 2b) during decoding. The

Cond. Inter-frame Coding	Residual Coding	Temporal Prior	BD-Rate
	✓		-21.8%
	✓	✓	-24.3%
✓			-28.9%
✓		✓	-33.1%

(a)

Input of Motion Coder	Cond.	BD-Rate
f_t	-	-33.4%
$f_t - f_c$	-	-35.3%
f_t	$\hat{f}_{t-1}$	-35.2%
f_t	f_c	-42.5%

(b)

Motion Coder	Inter-frame Coder	BD-Rate
DVC [26]	1-step CANF	-31.4%
DVC [26]	2-step CANF	-33.1%
1-step CANF	2-step CANF	-38.1%
2-step CANF	2-step CANF	-42.5%

(c)

Motion Coder	Inter-frame Coder	BD-Rate
1-step CANF	1-step CANF	35.3%
2-step CANF	2-step CANF	-42.5%
3-step CANF	3-step CANF	-31.4%

(d)

Table 3: (a) Comparison of conditional inter-frame coding and residual coding under the settings with and without the temporal prior. (b) Comparison of the conditional motion coding, predictive motion coding, and intra motion coding. (c)(d) Comparisons of the conditional motion and inter-frame coders with a varied number of autoencoding transforms. The rows with blue color are our proposed full model.

cascading effect, compounded by temporal error propagation, may outweigh the benefits of having more autoencoding transforms.

5   Conclusion

This work introduces CANF-VC for conditional inter-frame and motion coding. CANF-VC achieves the state-of-the-art video compression performance. Our major findings include: (1) the CANF-based inter-frame coding outperforms residual coding; (2) likewise, our conditional motion coding outperforms predictive motion coding at the cost of additional buffer requirements; (3) the quality of the conditioning variable is critical to compression performance; (4) our 2-step CANF performs better than 1-step CANF, justifying the use of multi-step CANF. Lastly, we note that CANF-VC does not use auto-regressive models in inter-frame and motion coding. Its operations are parallelizable.

Acknowledgements

References

1. Hm reference software for hevc. https://vcgit.hhi.fraunhofer.de/jvet/HM/-/tree/HM-16.20, accessed: 2022-03-03
2. Agustsson, E., Minnen, D., Johnston, N., Balle, J., Hwang, S.J., Toderici, G.: Scale-space flow for end-to-end optimized video compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8503–8512 (2020)
3. Ballé, J., Laparra, V., Simoncelli, E.P.: End-to-end optimized image compression. International Conference for Learning Representations (2017)
4. Ballé, J., Minnen, D., Singh, S., Hwang, S.J., Johnston, N.: Variational image compression with a scale hyperprior. In: International Conference on Learning Representations (2018)
5. Bégaint, J., Racapé, F., Feltman, S., Pushparaja, A.: Compressai: a pytorch library and evaluation platform for end-to-end compression research. arXiv preprint arXiv:2011.03029 (2020)
6. Brand, F., Seiler, J., Kaup, A.: Generalized difference coder: A novel conditional autoencoder structure for video compression. arXiv:2112.08011 (2021)
7. Bross, B., Wang, Y.K., Ye, Y., Liu, S., Chen, J., Sullivan, G.J., Ohm, J.R.: Overview of the versatile video coding (vvc) standard and its applications. IEEE Transactions on Circuits and Systems for Video Technology **31**(10), 3736–3764 (2021)
8. Chen, T., Liu, H., Ma, Z., Shen, Q., Cao, X., Wang, Y.: End-to-end learnt image compression via non-local attention optimization and improved context modeling. IEEE Transactions on Image Processing **30**, 3179–3191 (2021)
9. Cheng, Z., Sun, H., Takeuchi, M., Katto, J.: Learned image compression with discretized gaussian mixture likelihoods and attention modules. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7939–7948 (2020)
10. Dinh, L., Sohl-Dickstein, J., Bengio, S.: Density estimation using real nvp. Computing Research Repository (CoRR) (2016)
11. Frank, B.: Common test conditions and software reference configurations. JCTVC-L1100 **12**(7) (2013)
12. Golinski, A., Pourreza, R., Yang, Y., Sautiere, G., Cohen, T.S.: Feedback recurrent autoencoder for video compression. In: Proceedings of the Asian Conference on Computer Vision (2020)
13. Ho, Y.H., Chan, C.C., Peng, W.H., Hang, H.M., Domański, M.: Anfic: Image compression using augmented normalizing flows. IEEE Open Journal of Circuits and Systems **2**, 613–626 (2021)
14. Hu, Z., Chen, Z., Xu, D., Lu, G., Ouyang, W., Gu, S.: Improving deep video compression by resolution-adaptive flow coding. In: European Conference on Computer Vision. pp. 193–209. Springer (2020)
15. Hu, Z., Lu, G., Xu, D.: Fvc: A new framework towards deep video compression in feature space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1502–1511 (2021)
16. Huang, C.W., Dinh, L., Courville, A.: Augmented normalizing flows: Bridging the gap between generative flows and latent variable models. arXiv preprint arXiv:2002.07101 (2020)
17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. International Conference for Learning Representations (2015)

18. Kingma, D.P., Dhariwal, P.: Glow: Generative flow with invertible 1x1 convolutions. arXiv:1807.03039 (2018)
19. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
20. Kobyzev, I., Prince, S.J., Brubaker, M.A.: Normalizing flows: An introduction and review of current methods. IEEE transactions on Pattern Analysis and Machine Intelligence (2020)
21. Ladune, T., Philippe, P., Hamidouche, W., Zhang, L., Déforges, O.: Optical flow and mode selection for learning-based video coding. In: 2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP). pp. 1–6. IEEE (2020)
22. Ladune, T., Philippe, P., Hamidouche, W., Zhang, L., Déforges, O.: Conditional coding for flexible learned video compression. In: Neural Compression: From Information Theory to Applications–Workshop@ ICLR 2021 (2021)
23. Li, J., Li, B., Lu, Y.: Deep contextual video compression. Advances in Neural Information Processing Systems (2021)
24. Lin, J., Liu, D., Li, H., Wu, F.: M-lvc: multiple frames prediction for learned video compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3546–3554 (2020)
25. Liu, H., Lu, M., Ma, Z., Wang, F., Xie, Z., Cao, X., Wang, Y.: Neural video coding using multiscale motion compensation and spatiotemporal context model. IEEE Transactions on Circuits and Systems for Video Technology (2020)
26. Lu, G., Ouyang, W., Xu, D., Zhang, X., Cai, C., Gao, Z.: Dvc: An end-to-end deep video compression framework. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11006–11015 (2019)
27. Lu, G., Zhang, X., Ouyang, W., Chen, L., Gao, Z., Xu, D.: An end-to-end learning framework for video compression. IEEE transactions on Pattern Analysis and Machine Intelligence (2020)
28. Ma, H., Liu, D., Yan, N., Li, H., Wu, F.: End-to-end optimized versatile image compression with wavelet-like transform. IEEE Transactions on Pattern Analysis and Machine Intelligence (2020)
29. Mercat, A., Viitanen, M., Vanne, J.: Uvg dataset: 50/120fps 4k sequences for video codec analysis and development. In: Proceedings of the 11th ACM Multimedia Systems Conference. pp. 297–302 (2020)
30. Minnen, D., Ballé, J., Toderici, G.D.: Joint autoregressive and hierarchical priors for learned image compression. Advances in Neural Information Processing Systems **31**, 10771–10780 (2018)
31. Ranjan, A., Black, M.J.: Optical flow estimation using a spatial pyramid network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4161–4170 (2017)
32. Rippel, O., Anderson, A.G., Tatwawadi, K., Nair, S., Lytle, C., Bourdev, L.: Elf-vc: Efficient learned flexible-rate video coding. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 14479–14488 (October 2021)
33. Sullivan, G.J., Ohm, J.R., Han, W.J., Wiegand, T.: Overview of the high efficiency video coding (hevc) standard. IEEE Transactions on circuits and systems for video technology **22**(12), 1649–1668 (2012)
34. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8934–8943 (2018)

35. Wang, H., Gan, W., Hu, S., Lin, J.Y., Jin, L., Song, L., Wang, P., Katsavounidis, I., Aaron, A., Kuo, C.C.J.: Mcl-jcv: a jnd-based h. 264/avc video quality assessment dataset. In: 2016 IEEE International Conference on Image Processing (ICIP). pp. 1509–1513. IEEE (2016)
36. Xue, T., Chen, B., Wu, J., Wei, D., Freeman, W.T.: Video enhancement with task-oriented flow. International Journal of Computer Vision **127**(8), 1106–1125 (2019)
37. Yang, R., Mentzer, F., Gool, L.V., Timofte, R.: Learning for video compression with hierarchical quality and recurrent enhancement. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6628–6637 (2020)
38. Yang, R., Mentzer, F., Van Gool, L., Timofte, R.: Learning for video compression with recurrent auto-encoder and recurrent probability model. IEEE Journal of Selected Topics in Signal Processing **15**(2), 388–401 (2020)

CANF-VC: Conditional Augmented Normalizing Flows for Video Compression
Supplementary Materials

Yung-Han Ho[1], Chih-Peng Chang[1], Peng-Yu Chen[1], Alessandro Gnutti[2], and Wen-Hsiao Peng[1]

[1] Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan `wpeng@cs.nctu.edu.tw`

[2] Department of Information Engineering, CNIT, University of Brescia, Brescia, Italy `alessandro.gnutti@unibs.it`

This supplementary document provides additional materials to assist with the understanding of the performance and design of our CANF-VC. Specifically, it includes:

- CANF implementations
- Complexity characterization
- Rate-distortion curves with GOP 32
- Comparison with ELF-VC [10]
- Network details: CANF and motion extrapolation
- Temporal prior for motion coding
- Training strategy
- Subjective quality comparison
- Command lines for x265 and HM

1 CANF Implementations

Fig. A1 depicts two possible CANF implementations. Fig. A1a corresponds to the one presented in the main paper. It concatenates the motion-compensated reference frame x_c with the input frame x_t as input to all the encoding transforms. In comparison, the scheme in Fig. A1b additionally accepts x_c as input to all the decoding transforms. The former (decoding transforms w/o x_c) can be viewed as a special case of the latter (decoding transforms w/ x_c), which utilizes x_c for encoding transforms only. For the implementation of Fig. A1b, the latent code is decoded first to produce 16-channel features having the same spatial resolution as x_c. The resulting features are then concatenated with x_c before being processed further by the three convolution layers (the orange part in Fig. A1b) to complete the decoding transform. This implementation (Fig. A1b) has a slightly larger model size than Fig. A1a.

Table A1 presents the BD-rate comparison between the two CANF implementations. For experiments, we use the motion coder from DVC [7], while the inter-frame coder adopts the two different CANF implementations (Fig. A1a vs. Fig. A1b). It is seen that the more generalized implementation (decoding

Fig. A1: Illustration of CANF implementations: (a) Decoding transforms w/o x_c and (b) Decoding transforms w/ x_c.

Table A1: BD-rate comparison between two CANF implementations for conditional inter-frame coding. The motion coder is from DVC [7]. The anchor is x265 in veryslow mode.

Implementations	UVG [9]	MCL-JCV [12]	HEVC-B [11]
Decoding transforms w/o x_c	-33.1%	-15.3%	-35.4%
Decoding transforms w/ x_c	-35.2%	-15.3%	-33.9%

transforms w/ x_c) has comparable performance to our current implementation (decoding transforms w/o x_c) on all three datasets. This justifies our choice of decoding transforms w/o x_c because of its comparable performance and simpler design.

2   Complexity Characterization

Table A2 presents the computational characteristics of different competing methods from the perspectives of multiply-accumulate (MAC) operations, encoding/decoding times for inference, and model sizes. It is to be noted that the prolonged encoding/decoding times of DCVC [5] are due to the use of an auto-regressive model for entropy coding. Our CANF-VC neither uses an auto-regressive model for motion coding nor uses it for inter-frame coding. Its larger MAC

Table A2: Complexity characterization in terms of MACs, encoding/decoding times, and model sizes. The MACs and runtimes of DVC [7] and DCVC [5] are collected by running the test code released by the respective authors on the same 1080Ti platform. The MACs are evaluated based on encoding a 1080p P-frame, while the encoding/decoding times are averaged over the first 100 P-frames of the Beauty sequence in UVG dataset.

Method	MACs	Encoding/Decoding Time	Model Size
DVC [7]	1725G	4.15 s/4.06 s	8.5M
DVC_Pro [8]	-	-/-	29M
FVC [4]	-	-/-	26M
DCVC [5]	2268G	7.70 s/32.90 s	8M
CANF-VC Lite	4012G	1.38 s/0.98 s	15M
CANF-VC	5088G	1.60 s/1.05 s	31M

arises from stacking multiple autoencoding transforms. Nevertheless, its relatively short encoding/decoding times suggest that these autoencoding transforms are amenable to parallel computing. Lastly, we remark that the relatively longer encoding/decoding times of DVC [7] are due to their software implementation, particularly the entropy coding part. Our CANF-VC follows [2] to quantize the scale parameters from the hyperprior into 64 distinct values, enabling a fast table look-up to derive the probabilities for entropy coding. In contrast, DVC [7] does not quantize the scale parameters, and needs more time in evaluating higher-precision coding probabilities.

3   Rate-Distortion Curves with GOP Size 32

Fig. A2 presents rate-distortion curves for HM [1], DCVC [5], M-LVC [6], and our CANF-VC under GOP size 32 (see Table 2 for their BD-rate figures and Section 4.2 for detailed discussion). Except HM, all the competing methods use ANFIC [3] as the intra-frame coder for a fair comparison.

In terms of PSNR-RGB, our CANF-VC models outperform DCVC and M-LVC, except for CANF-VC Lite, which performs comparably to DCVC on MCL-JCV dataset. In addition, CANF-VC$^-$ shows worse performance than the other two CANF-VC variants at low rates because it does not include conditional motion coding, which is critical to low-rate compression performance. In comparison with HM, our CANF-VC shows better results at high rates, but worse results at low rates.

In terms of MS-SSIM-RGB, our CANF-VC models show slightly better results on UVG [9] and HEVC class B [11] than DCVC [5], and comparable results on MCL-JCV [12].

(a) UVG, PSNR-RGB

(b) UVG, MS-SSIM-RGB

(c) HEVC Class B, PSNR-RGB

(d) HEVC Class B, MS-SSIM-RGB

(e) MCL-JCV, PSNR-RGB

(f) MCL-JCV, MS-SSIM-RGB

Fig. A2: Comparison of rate-distortion curves on UVG, HEVC Class B, and MCL-JCV datasets for both PSNR and MS-SSIM. All the competing methods use ANFIC [3] as the intra-frame coder and are evaluated under the same setting, namely, 96-frame encoding with GOP size 32. Results for DCVC [5] and M-LVC [6] are produced by their released code.

4   Comparison with ELF-VC [10]

Table A3 presents separately the BD-rate comparison with ELF-VC [10] since ELF-VC [10] adopts a GOP size of 16, which is rarely used by the other competing methods. Note that the results of ELF-VC [10] are from their paper because its software is unavailable. Moreover, we note that ELF-VC [10] uses its own intra-frame coder, the details of which are unavailable. Under the same GOP

Table A3: BD-rate comparison under the same GOP size 16. The anchor is x265 in veryslow mode. Except ELF-VC [10], all the competing methods adopt ANFIC [3] as the intra-frame coder.

	BD-rate (%) PSNR-RGB		BD-rate (%) MS-SSIM-RGB	
	UVG	MCL-JCV	UVG	MCL-JCV
DCVC (ANFIC)	-19.5	-7.9	-47.4	-46.5
ELF-VC	-30.8	-11.7	-55.3	-53.9
CANF-VC Lite	-35.5	-12.0	-46.0	-44.4
CANF-VC-	-34.7	-13.5	-45.4	-43.9
CANF-VC	-41.0	-19.9	-50.3	-48.9

size and in terms of PSNR-RGB, we see that the superiority of our CANF-VC models to ELV-VC [10] and DCVC (with ANFIC as the intra-frame coder) is obvious. However, ELF-VC [10] achieves the best MS-SSIM-RGB results among all the competing methods. We remark that this comparison is to provide additional information; a fair comparison would require the software of ELF-VC [10] and more information about its intra-frame coder.

5   Network Details: CANF and Motion Extrapolation

Fig. A3 shows the network details of our CANF, where we choose $N = 128$ and $C = 128$, with M set to 192 for inter-frame coding and 128 for motion coding, respectively. Our CANF-VC Lite adopts $N = 72$ and $C = 128$, with $M = 128$ for both inter-frame and motion coding.

Fig. A4 depicts the network architecture of our U-Net-based motion extrapolation network.

6   Temporal Prior for Motion Coding

For conditional motion coding, our current implementation adopts the extrapolated image $warp(\hat{x}_{t-1}; f_c)$ for constructing the temporal prior (Section 3.3 of the main paper). Table A4 presents additional results for the case where the predicted flow map f_c is used instead. We see that the former achieves 8% more rate savings than the latter (i.e. using f_c to construct the temporal prior), which justifies our design choice. The reason may be that the flow map f_c is not as informative as $warp(\hat{x}_{t-1}; f_c)$, which contains more semantic and texture information.

7   Training Strategy

Table A5 summarizes our training steps in three major phases. The first phase uses uncompressed, original frames as inputs to the motion estimation and the

Fig. A3: Network details of our CANF-based coder.

Fig. A4: Network details of our U-Net-based motion extrapolation network.

motion extrapolation networks, in order to pre-train these networks. We then freeze them until the last three steps.

In the second (2-frame training) phase, we train the P-frame coder by encoding one P-frame with its reference frame being an uncompressed I-frame. In

Table A4: Comparison of different temporal priors for the motion coder.

Cond. Variable of Temporal Prior	BD-Rate (%)
$warp(\hat{x}_{t-1}; f_c)$	-42.5%
f_c	-34.4%

Table A5: Details of our training strategy.

Phase	Training Parts	Loss	lr	Epochs
Pre-training	motion estimation and motion extrapolation networks	D		
2-frame (IP) training	motion coder	$D+\lambda R$	1e-4	10
	motion coder and motion compensation network	$D+\lambda R$	1e-4	10
	Inter-frame coder	$D+\lambda R$	1e-4	8
	motion coder, motion compensation network, and inter-frame coder	$D+\lambda R$	1e-4	5
5-frame (IPPPP) training	motion coder, motion compensation network, and inter-frame coder	$D+\lambda R$	1e-4	5
	motion coder, motion compensation network, and inter-frame coder	$D+\lambda R$	5e-5	5
	All networks	$D+\lambda R$	2.5e-5	5
	All networks	$D+\lambda R$	5e-5	1
	All networks	$D+\lambda R$	2.5e-5	1

this phase, the uncompressed frames are used as inputs to the motion estimation and the motion extrapolation networks. We first train the motion coder and the motion compensation network. Subsequently, when the inter-frame coder is involved for training, we fix the motion coder and the motion compensation network for 8 epochs, followed by training jointly the inter-frame and the motion coders for another 5 epochs.

In the third (5-frame training) phase, we use 5 frames (IPPPP) as a basic training unit for forward propagation. However, in updating the P-frame coder, we stop the gradient at each reference frame so that the gradient will not back-propagate through reference frames. In this phase, the previously compressed frames are used as the reference frames (including I-frames) and are input to the motion estimation and the motion extrapolation networks. We train the inter-frame coder, the motion coder, and the motion compensation network for 10 epochs. Lastly, we fine-tune all the networks, including the motion estimation and the motion extrapolation networks, for another 7 epochs with learning rate decay.

8   Subjective Quality Comparison

Fig. A5 provides more subjective quality comparisons between CANF-VC and DCVC (ANFIC). Results are provided for models trained with PSNR and MS-SSIM. It is seen that our CANF-VC better preserves the shape of the objects than DCVC (ANFIC) (cf. the face of the jockey in the first row, the hair in the

Ground Truth	DCVC (ANFIC)	CANF-VC	DCVC-ssim (ANFIC)	CANF-VC-ssim

	PSNR-RGB: 33.84 dB 0.0184 bpp	PSNR-RGB: 34.50 dB 0.0109 bpp	MS-SSIM-RGB: 0.967 0.0336 bpp	MS-SSIM-RGB: 0.966 0.0271 bpp
	PSNR-RGB: 33.84 dB 0.0122 bpp	PSNR-RGB: 34.19 dB 0.0090 bpp	MS-SSIM-RGB: 0.956 0.0261 bpp	MS-SSIM-RGB: 0.955 0.0211 bpp
	PSNR-RGB: 28.09 dB 0.0697 bpp	PSNR-RGB: 29.02 dB 0.0638 bpp	MS-SSIM-RGB: 0.957 0.0739 bpp	MS-SSIM-RGB: 0.959 0.0704 bpp
	PSNR-RGB: 32.68 dB 0.0343 bpp	PSNR-RGB: 33.26 dB 0.0267 bpp	MS-SSIM-RGB: 0.972 0.0506 bpp	MS-SSIM-RGB: 0.970 0.0390 bpp
	PSNR-RGB: 33.96 dB 0.0265 bpp	PSNR-RGB: 35.50 dB 0.0232 bpp	MS-SSIM-RGB: 0.979 0.0348 bpp	MS-SSIM-RGB: 0.981 0.0369 bpp

Fig. A5: Subjective quality comparison between CANF-VC and DCVC (ANFIC). The suffix "-ssim" indicates that the models are trained with MS-SSIM.

second row, the letters "DE" in the third row, the necklace in the forth row, and the textured pattern in the last row).

9   Command Lines for X265 and HM

Following [5], we use FFmpeg to generate the compressed videos through x265 with veryslow mode. Given an uncompressed video "input.yuv" of size W × H, the command line for x265 encoding is as follows: *ffmpeg -pix fmt yuv420p -s W ×H -r FR -i input.yuv -vframes N -c:v libx265 -preset veryslow -tune ze- rolatency -x265-params "qp=Q:keyint=GOP:verbose=1" output.mkv*, where FR, N, Q, GOP represent the frame rate, the number of frames to be encoded, the quantization parameter and the GOP size, respectively. Q is set to 19, 22, 27, 32, 37. For the common test protocol, we choose GOP to be 10 for HEVC Class B and 12 for the other datasets.

For the encoding with HM, given an uncompressed video "input.yuv" of size W × H, we use the *encoder_lowdelay_P_main.cfg* configuration file [1] with the following parameters: InputFile=input.yuv, FrameRate=FR, SourceWidth=W, SourceHeight=H, FramesToBeEncoded=N, IntraPeriod=32, GOPSize=8, De- codingRefreshType=2, and QP=Q, where FR, N, Q represent the frame rate, the number of frames to be encoded, and the quantization parameter, respec- tively. Q is set to 17, 22, 24, 27, 32.

References

1. Hm reference software for hevc. https://vcgit.hhi.fraunhofer.de/Zhu/HM/- /blob/HM-16.22/cfg/encoder_lowdelay_P_main.cfg, accessed: 2022-03-10
2. Ballé, J., Minnen, D., Singh, S., Hwang, S.J., Johnston, N.: Variational image compression with a scale hyperprior. In: International Conference on Learning Representations (2018)
3. Ho, Y.H., Chan, C.C., Peng, W.H., Hang, H.M., Domański, M.: Anfic: Image com- pression using augmented normalizing flows. IEEE Open Journal of Circuits and Systems **2**, 613–626 (2021)
4. Hu, Z., Lu, G., Xu, D.: Fvc: A new framework towards deep video compression in feature space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1502–1511 (2021)
5. Li, J., Li, B., Lu, Y.: Deep contextual video compression. Advances in Neural Information Processing Systems (2021)
6. Lin, J., Liu, D., Li, H., Wu, F.: M-lvc: multiple frames prediction for learned video compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3546–3554 (2020)
7. Lu, G., Ouyang, W., Xu, D., Zhang, X., Cai, C., Gao, Z.: Dvc: An end-to-end deep video compression framework. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11006–11015 (2019)
8. Lu, G., Zhang, X., Ouyang, W., Chen, L., Gao, Z., Xu, D.: An end-to-end learn- ing framework for video compression. IEEE transactions on Pattern Analysis and Machine Intelligence (2020)
9. Mercat, A., Viitanen, M., Vanne, J.: Uvg dataset: 50/120fps 4k sequences for video codec analysis and development. In: Proceedings of the 11th ACM Multimedia Systems Conference. pp. 297–302 (2020)

10. Rippel, O., Anderson, A.G., Tatwawadi, K., Nair, S., Lytle, C., Bourdev, L.: Elf-vc: Efficient learned flexible-rate video coding. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 14479–14488 (October 2021)
11. Sullivan, G.J., Ohm, J.R., Han, W.J., Wiegand, T.: Overview of the high efficiency video coding (hevc) standard. IEEE Transactions on circuits and systems for video technology **22**(12), 1649–1668 (2012)
12. Wang, H., Gan, W., Hu, S., Lin, J.Y., Jin, L., Song, L., Wang, P., Katsavounidis, I., Aaron, A., Kuo, C.C.J.: Mcl-jcv: a jnd-based h. 264/avc video quality assessment dataset. In: 2016 IEEE International Conference on Image Processing (ICIP). pp. 1509–1513. IEEE (2016)