Hadizadeh and Bajić
EURASIP Journal on Image and Video Processing (2024) 2024:41
https://doi.org/10.1186/s13640-024-00657-w

EURASIP Journal on Image and Video Processing

RESEARCH Open Access

Learned scalable video coding for humans and machines

Hadi Hadizadeh[1*] and Ivan V. Bajić[1]

A Python implementation of the proposed system is available at https://github.com/hadipardis/svc.

*Correspondence: hadi_hadizadeh@sfu.ca

[1] School of Engineering Science, Simon Fraser University, 8888 University Drive, Burnaby, BC V5A 1S6, Canada

Abstract

Video coding has traditionally been developed to support services such as video streaming, videoconferencing, digital TV, and so on. The main intent was to enable human viewing of the encoded content. However, with the advances in deep neural networks (DNNs), encoded video is increasingly being used for automatic video analytics performed by machines. In applications such as automatic traffic monitoring, analytics such as vehicle detection, tracking and counting, would run continuously, while human viewing could be required occasionally to review potential incidents. To support such applications, a new paradigm for video coding is needed that will facilitate efficient representation and compression of video for both machine and human use in a scalable manner. In this manuscript, we introduce an end-to-end learnable video codec that supports a machine vision task in its base layer, while its enhancement layer, together with the base layer, supports input reconstruction for human viewing. The proposed system is constructed based on the concept of conditional coding to achieve better compression gains. Comprehensive experimental evaluations conducted on four standard video datasets demonstrate that our framework outperforms both state-of-the-art learned and conventional video codecs in its base layer, while maintaining comparable performance on the human vision task in its enhancement layer.

Keywords: Video compression, Video analytics, Scalable coding, Deep learning, Coding for machines

1 Introduction

Video analytics is a crucial technology that finds application in a wide range of fields, such as visual surveillance, traffic monitoring, and autonomous navigation. In many applications, input video is compressed and transmitted to the cloud for subsequent analysis. Depending on the specific application, there are several approaches that can be adopted. When machine-based video analytics is the sole focus, precomputed features can be compressed and transmitted to the cloud instead of the complete video, thus reducing the required bandwidth [1]. Various hand-crafted and neural network-based (learned) approaches have been developed to serve this purpose [2, 3]. However, if human viewing is also required, the input video must also be encoded and transmitted, increasing the complexity of the system.

Springer Open

© The Author(s) 2024. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

Another approach involves compressing the input video and transmitting it to the cloud, where it will be decoded and used for visual analysis or human viewing. Various video codecs have been proposed over the last few decades for this purpose, including high efficiency video coding (HEVC) [4] and versatile video coding (VVC) [5]. However, if human viewing is needed only occasionally, this approach is inefficient, as will be shown later in the paper.

The swift progress in deep learning technologies has led to the emergence of several learned image and video codecs that rely on deep neural networks (DNNs) to compete with established conventional counterparts like JPEG, HEVC, and VVC. Notable methods include [6–14]. Nevertheless, a majority of DNN-based codecs have focused on image and video compression for human viewing. On the other hand, DNNs are widely used in visual analytics, but their primary emphasis is not on compression [15]. An important recent development in this space is JPEG AI [16], a DNN-based image coding standard whose aim is to support both human viewing and machine analysis within a single unified bitstream. An advantage of JPEG AI over earlier coding standards is that it offers the opportunity to bypass input image reconstruction and feed entropy-decoded latent representation into a DNN-based back-end for analysis, thereby facilitating potential savings in terms of complexity. However, the JPEG AI standard itself does not offer the means for the encoder to distinguish machine task-relevant from task-irrelevant information. Hence, when human viewing is needed only occasionally, this approach faces similar challenges in terms of bit-efficiency as earlier coding standards, because task-irrelevant information is encoded into the bitstream.

Recently, new video standardization efforts—MPEG-VCM (video coding for machines) [17] and MPEG-FCM (feature coding for machines)—have been initiated with the aim of establishing a standardized framework for video coding that caters to the needs of both human and machine vision. Various approaches have also been proposed for scalable image coding for multiple tasks [18–20], including machine vision and human viewing. In these methods, the base layer usually supports the machine vision task (e.g., object detection) while the enhancement layer supports input reconstruction for human viewing.

As discussed in [20], a typical machine vision pipeline involves image encoding, decoding, and the analysis of the decoded image by a machine vision model. This process can be represented by a Markov chain [21] $X \rightarrow Y \rightarrow \widehat{X} \rightarrow V$, where X denotes the input image, Y represents the latent representation of X, $\widehat{X}$ is the reconstructed image, and V is the output of the machine vision model. Applying the data processing inequality [21] to this chain, we can infer that $I(Y;\widehat{X}) \geqslant I(Y;V)$, where $I(\cdot;\cdot)$ represents the mutual information [21]. This suggests that machine vision may require less information (fewer bits, if optimally encoded) than input reconstruction. This aligns with scalable human–machine coding approaches mentioned above, where both base and enhancement layers are utilized for input reconstruction, but only the base layer (i.e. fewer bits) is used for machine vision.

In this paper, we present an end-to-end learned scalable human–machine video codec. The proposed codec is designed to support object detection in its base layer, and input reconstruction by using both base and enhancement layer. The proposed system utilizes the back-end of a well-known DNN-based object detection network (YOLOv5) [22] to

execute the object detection task through its base layer. However, the architecture of the proposed system is versatile, and it can be integrated with other machine vision models. Since object detection is a fundamental machine vision task that serves as a building block for more complex video analytics—such as video surveillance, traffic monitoring, object tracking, smart home, security systems, and so on—it was a suitable choice that serves two purposes: (1) it provides information for a variety of downstream analytics applications, and (2) it allows us to demonstrate the compression benefits of removing task-irrelevant information from the base layer, thereby achieving significant gains over conventional approaches when only machine analysis is required. In terms of compression, the proposed system is constructed based on the notion of conditional coding [9, 10, 14, 23], which is theoretically more efficient than the widely employed residual coding, as discussed in Sect. 3.

We motivate our codec design by analyzing the efficiency of lossy conditional coding compared to lossy residual coding. In Sect. 3, we present rate-distortion analysis of lossy conditional coding. To our knowledge, such analysis has not been presented in the video coding literature thus far. Our proposed scalable video coding system consists of two layers: a base layer and an enhancement layer. For the base layer, we leveraged our previous work presented in [23]. For the enhancement layer, we incorporated some modules from the codec introduced in [14] and added a novel feature extraction and fusion model, as well as a new entropy model, to enable efficient coding of the enhancement layer conditioned on the base layer.

After building a compression system on the principles of conditional coding, we performed extensive experiments to assess its performance. Our experiments on four standard video datasets demonstrate that the base layer of the proposed system outperforms state-of-the-art conventional and learned video codecs in coding for object detection, while the enhancement layer provides comparable performance for human viewing to those obtained by the existing codecs. In fact, our proposed codec serves as a practical demonstration of the effectiveness of conditional coding. The proposed codec is well suited for applications where automated machine vision needs to run continuously while human viewing is needed occasionally, which we demonstrate through break-even analysis.

This paper is organized as follows: we review some of the related prior work on learned video coding for humans and machines, and discuss the differences relative to our present work in Sect. 2. After that, we provide a rate-distortion analysis of conditional coding in Sect. 3, which is used in our codec design. The proposed system is presented in Sect. 4. The experimental results are provided in Sect. 5, and the conclusions are drawn in Sect. 6.

2 Related work

In recent years, a number of learned video codecs have emerged in the literature. Lu *et al.* [24] introduced Deep Video Compression (DVC), the first end-to-end learned video codec utilizing temporal predictive coding. Agustsson *et al.* [13] proposed an end-to-end video coding model based on a learning-driven motion compensation framework, where a warped frame generated by a learned flow map serves as the predictor for coding the current video frame. Liu *et al.* [25] employed feature-domain warping in a coarse-to-fine

approach for video compression, while Hu *et al.* [26] leveraged deformable convolutions for feature warping.

Most traditional video codecs rely on residual coding. However, Ladune *et al.* [12, 27] argued that conditional coding relative to a predictor is more efficient than residual coding when using the same predictor. Building on this concept, Ho *et al.* [10] introduced Conditional Augmented Normalizing Flows for Video Coding (CANF-VC), a high-performance learned video codec that utilizes conditional coding for both motion and inter-frame coding. Building on CANF-VC's conditional coding engines, we developed a codec called Learned Conditional Coding Modes for Video Coding (LCCM-VC) [23], which incorporates additional learned coding modes to further enhance compression performance. In [14], a method called DCVC-TCM was proposed, utilizing a set of temporal multi-scale contexts as predictors for conditional coding of video frames.

The methods mentioned above were specifically designed for video coding for human viewing. However, a few learned scalable video coding techniques have been developed that cater to both human vision and machine analytics. For example, [28] presents a scalable human–machine video codec based on the concept of latent-space scalability [29]. This codec was designed using a hybrid approach that leverages DNN-based compression for intra-coding, and a combination of DNN and conventional techniques for inter-frame coding. In [30], a human-centric video compression system for human and machines was proposed, which can restore faces with high perceptual quality for human viewing at very low bitrates while maintaining high recognition accuracy for machine analytics. A human–machine friendly video compression scheme (HMFVC) was proposed in [31]. This scheme supports action recognition in its semantic (base) layer and input reconstruction for human viewing in its enhancement layer. A deep scalable video coding for human and machine vision, called DeepSVC, was proposed in [32], which supports a semantic layer for object detection, a structure layer for low-bitrate input reconstruction, and a texture layer for higher quality input reconstruction.

The proposed system differs from [28] in several ways. First, [28] is a hybrid codec that incorporates both DNN-based and conventional handcrafted video coding tools, while the system presented in this paper is an end-to-end learned codec. Second, [28] used only intra-coding in the base layer, while our proposed system utilizes inter-frame coding for both the base and enhancement layers. Third, [28] employed conventional residual inter-frame coding in the enhancement layer, while our proposed codec is built upon conditional coding. Finally, [28] used YOLOv3 [33] for object detection in the base layer while our codec uses the more recent YOLOv5 [22]. The difference with respect to [31] is twofold. First, the machine task in [31] is action recognition while ours is object detection. Second, [31] is built upon residual coding whereas our codec utilizes conditional coding. The proposed system also differs from [30] in that [30] is designed specifically for face videos (e.g., videoconferencing), whereas our codec is applicable across diverse application scenarios. Finally, although our codec and [32] both consider object detection as the machine task, there are several key differences. First, [32] utilizes TROIAlign [34] for object detection, whereas our codec is based on the popular YOLOv5 pipeline. Second, there are architectural differences in both the coding engines and the layer structure. Finally, unlike [32], which applies conditional coding solely for enhancement, we employ conditional coding in both base and enhancement layers.

3 Rate-distortion analysis of conditional coding

In this section, we provide motivation for our compression approach by analyzing residual and conditional coding from the rate-distortion point of view. We will use capital letters (X) to denote random variables and lowercase letters (x) for their realizations.

Conditional coding has been argued in [9, 10, 12, 14, 23] to offer the potential for better compression performance compared to residual coding. However, these studies offered only arguments based on lossless compression. Specifically, if X is the input to be compressed and Y is its predictor, then it can be shown [23] that

$$H(X|Y) \leqslant H(X-Y), \tag{1}$$

where $H(\cdot)$ is the entropy and $H(\cdot\,|\,\cdot)$ is the conditional entropy [21]. If we were interested in lossless coding, this would be enough to motivate the choice of a conditional codec over a residual one, since entropy is the limit of lossless compression. However, most practical codecs are lossy, so we need to employ rate-distortion theory to argue in favor of conditional versus residual coding. In this section, we offer such analysis, which can be considered as the extension of (1) into the realm of rate-distortion.

First we need to recall a few concepts from rate-distortion theory [21]. Let $d(x,\hat{x})$ be a distortion measure between an input x and its reconstruction $\hat{x}$. The expected distortion between input X and its reconstruction $\widehat{X}$ is computed as:

$$\mathbb{E}\left[d(X,\widehat{X})\right] = \sum_{x,\hat{x}} p_X(x)\cdot p_{\widehat{X}|X}(\hat{x}|x)\cdot d(x,\hat{x}), \tag{2}$$

where $p_X(x)$ is the input distribution and $p_{\widehat{X}|X}(\hat{x}|x)$ is the conditional distribution of the reconstruction given the input, also known as the *quantizer*. The input distribution is not under our control, but the quantizer is. For a given distortion level $D>0$, the set of feasible quantizers for X that achieve expected distortion of at most D is denoted

$$\mathcal{P}_X(D) = \left\{p_{\widehat{X}|X}(\hat{x}|x)\ :\ \mathbb{E}\left[d(X,\widehat{X})\right] \leqslant D\right\}. \tag{3}$$

The rate distortion function $R(D)$ is the minimum achievable rate for encoding input X while incurring expected distortion of at most D. It is given by:

$$R(D) = \min_{p_{\widehat{X}|X}(\hat{x}|x)\in\mathcal{P}_X(D)} I(X;\widehat{X}), \tag{4}$$

where $I(\cdot;\cdot)$ is the mutual information [21]. Note that in rate-distortion theory, rate is governed by mutual information rather than entropy, which is why (1) cannot be used directly to motivate conditional lossy compression.

Consider the two coding schemes illustrated in Fig. 1, where the top part shows residual coding and the bottom part shows conditional coding. In residual coding, prediction Y is subtracted from the input X to create $Z=X-Y$, which is then encoded and reconstructed as $\widehat{Z}$. Finally, input reconstruction is obtained as $\widehat{X}=\widehat{Z}+Y$. If the distortion is shift-invariant[1] then $d(x,\hat{x}) = d(x-y,\hat{x}-y) = d(z,\hat{z})$, so $\mathbb{E}[d(X,\widehat{X})] = \mathbb{E}[d(Z,\widehat{Z})]$. Hence, the rate-distortion function for residual coding of X is

[1] Many conventional distortion measures, including norm-based measures, are shift-invariant.

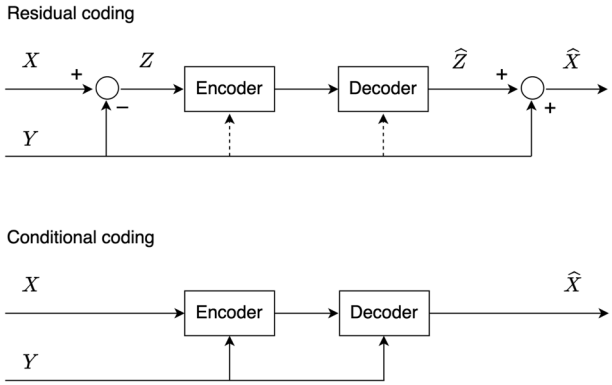


Fig. 1 Top: residual coding. Dashed lines are not used in residual coding, they are shown to aid the analysis in the text. Bottom: conditional coding

$$R_r(D) = \min_{p_{\widehat{Z}|Z}(\hat{z}|z)\in\mathcal{P}_Z(D)} I(Z;\widehat{Z}), \tag{5}$$

where subscript r indicates residual coding and

$$\mathcal{P}_Z(D) = \left\{ p_{\widehat{Z}|Z}(\hat{z}|z) \,:\, \mathbb{E}\left[d(X,\widehat{X})\right] \leqslant D \right\}. \tag{6}$$

Meanwhile, by applying (4) to the input (X, Y) and considering the case where the distortion is 0 everywhere on the support of Y, we arrive at the rate-distortion function for conditional coding of X relative to Y (Fig. 1 bottom):

$$R_c(D) = \min_{p_{\widehat{X}|X,Y}(\hat{x}|x,y)\in\mathcal{P}_{X|Y}(D)} I(X;\widehat{X}|Y), \tag{7}$$

where subscript c indicates conditional coding, $I(\cdot;\cdot|\cdot)$ is the conditional mutual information [21], and

$$\mathcal{P}_{X|Y}(D) = \left\{ p_{\widehat{X}|X,Y}(\hat{x}|x,y) \,:\, \mathbb{E}\left[d(X,\widehat{X})\right] \leqslant D \right\}. \tag{8}$$

We expand $I(X;\widehat{X}|Y)$ as follows:

$$\begin{aligned} I(X;\widehat{X}|Y) &= H(X|Y) - H(X|\widehat{X},Y) \\ &\overset{(a)}{=} H(X-Y|Y) - H(X-Y|\widehat{X},Y) \\ &= I(X-Y;\widehat{X}|Y) \\ &= H(\widehat{X}|Y) - H(\widehat{X}|Y,X-Y) \\ &\overset{(b)}{=} H(\widehat{X}-Y|Y) - H(\widehat{X}-Y|Y,X-Y) \\ &= I(X-Y;\widehat{X}-Y|Y) \\ &\overset{(c)}{=} I(Z;\widehat{Z}|Y), \end{aligned} \tag{9}$$

where (a) (resp. (b)) follows from the fact that given Y, the only uncertainty in $X - Y$ (resp. $\widehat{X} - Y$) is due to X (resp. $\widehat{X}$), and (c) follows from the fact that $Z = X - Y$ and $\widehat{Z} = \widehat{X} - Y$. Moreover, due to these relationships, it should be clear that conditioned on Y, minimization over $p_{\widehat{X}|X,Y}(\hat{x}|x,y)$ is equivalent to minimization over $p_{\widehat{Z}|Z,Y}(\hat{z}|z,y)$. Combining this with (9), we have that $R_c(D)$ from (7) can also be written as:

$$R_c(D) = \min_{p_{\widehat{Z}|Z,Y}(\hat{z}|z,y) \in \mathcal{P}_{Z|Y}(D)} I(Z; \widehat{Z}|Y), \tag{10}$$

where

$$\mathcal{P}_{Z|Y}(D) = \left\{ p_{\widehat{Z}|Z,Y}(\hat{z}|z,y) \ : \ \mathbb{E}\left[d(X, \widehat{X}) \right] \leqslant D \right\}. \tag{11}$$

Hence, conditional coding of X relative to Y achieves the same rate distortion as conditional coding of Z relative to Y. This is shown in Fig. 1 (top), where the dashed line indicates conditional coding, mimicking Fig. 1 (bottom).

Next, we turn to the seminal paper on conditional rate-distortion [35]. In the notation used in [35], we have that our $R_r(D) = R_Z(D)$ and our $R_c(D) = R_{Z|Y}(D)$. Theorem 3.1 in [35] shows that $R_{Z|Y}(D) \leqslant R_Z(D)$, with equality if and only if Y and Z are independent. Hence, we also have that

$$R_c(D) \leqslant R_r(D), \tag{12}$$

meaning that for any given distortion level $D > 0$, optimal conditional coding achieves the rate no higher than optimal residual coding. According to [35], the equality in (12) would hold if and only if Y and $Z = X - Y$ were independent, i.e., if Y and X were independent. Hence, so long as the side information Y provides at least some information about X, (12) becomes a strict inequality. In practice, however, we are generally dealing with sub-optimal codecs, so (12) might not always hold for practical codecs.

Equation (12) is the rate-distortion analog of (1). In fact, (1) provides the result for the special case $D = 0$ (lossless coding), while (12) extends this to $D > 0$. Intuition behind it is shown in Fig. 1 (top), where the dashed lines imply that the conditional codec for Z can, at the very least, ignore Y, and thereby match any performance that residual codec can achieve.

4 Proposed method

In this section, we present our proposed scalable video coding system, which is composed of a base layer and an enhancement layer. We first present an overview of the proposed codec in Sect. 4.1. We then describe the structure of the base layer in Sect. 4.2, followed by the structure of the enhancement layer in Sect. 4.3. The overall block diagram of the proposed system is depicted in Fig. 2.

4.1 Overview

Let $x_{1:T} \in \mathbb{R}^{T \times 3 \times H \times W}$ denote a raw (RGB) video sequence of T frames $\{x_t,\ t = 1, \cdots, T\}$ each of width W and height H. The aim is to design an end-to-end learnable system to code the input video in a scalable manner that caters to both machine vision and human viewing. We design a system whose base layer facilitates object detection, intended for

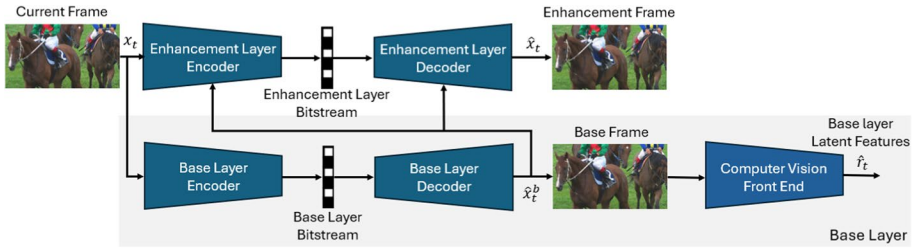


Fig. 2 The block diagram of the proposed learned scalable video compression system. The input video frame is x_t. The base layer has two outputs: the base frame $\hat{x}_t^b$ and the latent features $\hat{r}_t$ by which the computer vision task is performed. The enhancement layer has only one output, enhancement frame, which is the reconstructed frame $\hat{x}_t$

machine usage, while the enhancement layer, together with the base layer, facilitates reconstruction of the input video for human viewing. Although our scalable system is designed to support object detection, it can be re-trained to support any other machine vision task in the base layer. We chose object detection because it forms the basis for more complex machine vision scenarios such as video surveillance, tracking, traffic monitoring, etc. In our design, we exploit the concept of conditional coding which, as shown in the previous section, holds a theoretical advantage over the more common residual coding. In summary, the proposed system operates as follows:

- The input video frame is initially coded by the base layer to generate a reconstructed (decoded) base frame.
- The reconstructed base frame is then utilized by a computer vision model to perform the designated machine vision task (e.g., object detection).
- Simultaneously, the reconstructed base frame serves as a predictor in the enhancement layer for conditional coding of the input video frame, resulting in a reconstructed frame optimized for human viewing.

Specifically, the base layer encoder codes every video frame, x_t, into a compressed base layer bitstream. The base layer decoder then decodes the compressed bitstream to obtain a decoded RGB frame, $\hat{x}_t^b$, also referred to as the decoded *base frame*. This frame is utilized by a computer vision (CV) model (in our case YOLOv5 [22]) for video analysis (in our case object detection). Furthermore, $\hat{x}_t^b$ also helps the enhancement layer perform the reconstruction of the input frame for human viewing. The base layer is trained to maximize object detection accuracy while minimizing the rate of the base layer's bitstream. Note that the output of the base layer is an RGB frame. We designed it this way to ensure seamless integration with any existing computer vision network in a plug-and-play manner, eliminating the need for architectural modifications. For instance, this design enables us to easily replace YOLO [33] with any object detector, such as SSD [36], without changing their input architecture. These detectors expect an RGB image in their input, and our base layer outputs an RGB image.

The input video frame, x_t, is also fed into the enhancement layer alongside the base frame, $\hat{x}_t^b$. The enhancement layer's encoder then compresses the input frame conditionally to generate a compressed bitstream, which the enhancement layer's decoder utilizes to reconstruct an output frame, $\hat{x}_t$. The enhancement layer is trained end-to-end

to minimize both the distortion between x_t and $\hat{x}_t$, as well as the rate of the enhancement layer's bitstream. In the proposed system, the base layer is trained first, followed by training the enhancement layer while keeping the base layer frozen. In scalable human–machine image coding, this strategy has recently been shown to be advantageous over parallel training of the base and enhancement layer [37], so we use it for our proposed video codec as well.

It is important to note that in practice, if the sole objective is to perform the computer vision task, there is no need to reconstruct the input frame. That is, the enhancement layer's bitstream does not need to be created because $\hat{x}_t$ does not need to be reconstructed at the decoder. If human viewing is required, the enhancement layer can be employed and $\hat{x}_t$ reconstructed. Many video analytics applications, such as traffic monitoring or video surveillance, require continuous use of the analytics engine (base layer in our system), and only occasional human viewing (enhancement layer), for cases like emergencies or incident review.

4.2 The base layer

As shown in Fig. 3, the base layer consists of several components, which are described below. The base layer codec is built upon our previous work LCCM-VC [23]; the main difference is that the codec in [23] was trained to preserve the fidelity of the input frame under a rate constraint, while our base layer codec here is trained to preserve task-relevant features $\hat{r}_t$ under a rate constraint.

4.2.1 Motion estimation, extrapolation, and coding

Initially, the input video frame, x_t, and the previous base frame, $\hat{x}^b_{t-1}$, are fed into a DNN-based motion estimation network. This network is designed to estimate the optical flow, denoted by v^b_t, between x_t and $\hat{x}^b_{t-1}$. For this purpose, we use PWC-Net [38]. The flow v^b_t is then encoded by a DNN-based encoder, V_π, to obtain a compressed motion bitstream in the base layer. This stream is decoded by a DNN-based decoder, V_π^{-1}, to obtain

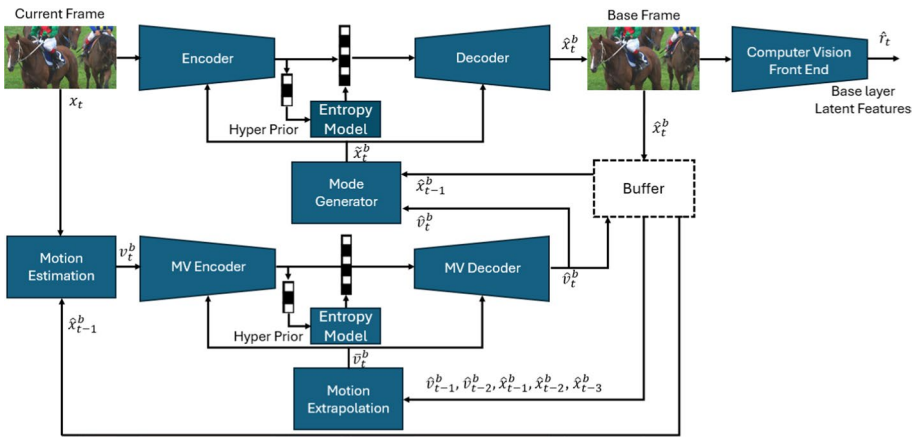


Fig. 3 The block diagram of the base layer of the proposed system. The base layer has two outputs: the reconstructed base frame $\hat{x}^b_t$ and the latent features $\hat{r}_t$ by which the computer vision task is performed. Our base layer's architecture is the same as our previous work LCCM-VC [23]. The mode generator is explained in greater detail in Fig. 4

a decoded flow, $\hat{v}_t^b$. For this purpose, we adopt the conditional coder from CANF-VC [10], which codes a signal conditioned on its predictor to achieve better rate-distortion performance. To obtain a predictor for conditional coding of v_t^b, we utilized the motion extrapolation network proposed in [10]. This network leverages the two previously decoded flow maps, $\hat{v}_{t-2}^b$ and $\hat{v}_{t-1}^b$, as well as the three previously decoded base frames, $\hat{x}_{t-3}^b$, $\hat{x}_{t-2}^b$, and $\hat{x}_{t-1}^b$, to generate a predictor, $\bar{v}_t^b$, for v_t^b.

The optical flow map $\hat{v}_t^b$ is used to warp the previously decoded base frame $\hat{x}_{t-1}^b$ through a motion compensation network to obtain a predictor frame, $\bar{x}_t^b$, for conditional coding of the current frame x_t. Also, $\hat{v}_t^b$ is stored in a flow buffer to be used for motion extrapolation of the subsequent frames. We used the same motion compensation network as proposed in [10].

4.2.2 Mode generator

To improve the coding performance of the base layer, we used the idea of learned conditional coding modes proposed in our earlier work on learned video coding, called LCCM-VC [23]. Specifically, the previously reconstructed base frame $\hat{x}_{t-1}^b$, the motion-compensated frame $\bar{x}_t^b$, and the decoded flow $\hat{v}_t^b$ are fed to the mode generator from [23] to obtain two weight maps, $\hat{\alpha}_t^b$ and $\hat{\beta}_t^b$. Since these maps are produced from previously (de)coded data, they can be regenerated at the decoder without any additional bits. As shown in Fig. 4, using $\hat{\beta}_t^b$, an enhanced predictor, $\tilde{x}_t^b$, is generated as follows:

$$\tilde{x}_t^b = \hat{\beta}_t^b \odot \bar{x}_t^b + (\mathbb{1} - \hat{\beta}_t^b) \odot \hat{x}_{t-1}^b, \tag{13}$$

where $\odot$ denotes Hadamard (element-wise) product and $\mathbb{1}$ is the all-ones matrix. At each pixel location, the value of $\tilde{x}_t^b$ is a weighted combination of $\bar{x}_t^b$ and $\hat{x}_{t-1}^b$, where the weight is defined by the corresponding value in $\hat{\beta}_t^b$. Hence, $\tilde{x}_t^b$ is a predictor for the current frame being coded, whose values are adaptively obtained from the previously reconstructed base frame ($\hat{x}_{t-1}^b$) and the motion-compensated base frame ($\bar{x}_t^b$). After that, $\tilde{x}_t^b$ is multiplied by $\hat{\alpha}_t^b$ in a pixel-wise manner, and the obtained result, i.e. $\hat{\alpha}_t^b \odot \tilde{x}_t^b$, is fed as a predictor to the inter-frame coder described next. Note that, using (13), the enhanced predictor can be considered as a linear combination of $\bar{x}_t^b$ and $\hat{x}_{t-1}^b$. When $\hat{\beta}_t^b \to 0$, it becomes equal to $\hat{x}_{t-1}^b$, and when $\hat{\beta}_t^b \to \mathbb{1}$, it becomes equal to $\bar{x}_t^b$.

4.2.3 Base inter-frame coder

The predictor $\hat{\alpha}_t^b \odot \tilde{x}_t^b$ is used for conditional coding of the current frame x_t. For this purpose, we utilized LCCM-VC [23], which is an extension of CANF-VC [10]. Using this framework, the conditional encoder of the base inter-frame coder, G_π, encodes $\hat{\alpha}_t^b \odot x_t$

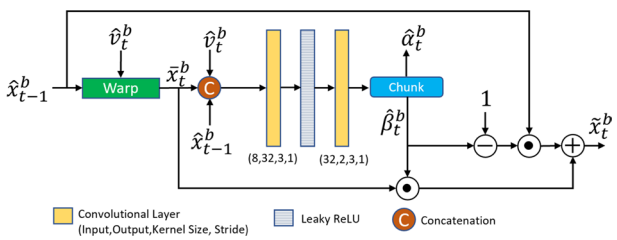


Fig. 4 The structure of the mode generator

using $\hat{\alpha}_t \odot \tilde{x}_t^b$ to produce a compressed bitstream, which is used to decode an image $\check{x}_t^b$ via G_π^{-1}. Finally, the reconstructed base frame, $\hat{x}_t^b$, is generated as follows:

$$\hat{x}_t^b = \check{x}_t^b + (\mathbb{1} - \hat{\alpha}_t^b) \odot \tilde{x}_t^b. \tag{14}$$

Note that, for pixels where $\hat{\alpha}_t^b \to 0$, $\hat{x}_t$ becomes equal to $\tilde{x}_t$, so the inter-frame coder does not need to code anything. This resembles the SKIP mode in conventional video coders, and depending on the value of $\hat{\beta}_t^b$, the system can copy directly from $\hat{x}_{t-1}^b$, $\bar{x}_t^b$, or a mixture of these two, to obtain $\tilde{x}_t^b$. When $\hat{\alpha}_t^b \to \mathbb{1}$, only the inter-frame coder is used to obtain $\hat{x}_t$. In the limiting case when $\hat{\alpha}_t^b \to \mathbb{1}$ and $\hat{\beta}_t^b \to \mathbb{1}$, LCCM-VC [23] becomes CANF-VC [10]. Hence, as discussed in [23], LCCM-VC has more flexibility and a larger number of conditional coding modes than CANF-VC. The architecture of G_π and G_π^{-1} is similar to that of V_π and V_π^{-1}.

4.2.4 Task-relevant feature extraction

The obtained base frame, $\hat{x}_t^b$, is stored in the base frame buffer for subsequent frame coding and is also fed to the base task network. In our work, this is YOLOv5 [22] for object detection, but in principle any (differentiable) network could be used here for any computer vision task. To enable unsupervised training of the base layer, we truncate the task network and utilize a copy of the resulting front-end of the network within the base layer, as described next. Specifically, we feed $\hat{x}_t^b$ to the cloned front-end network in the following manner:

$$\hat{r}_t = F_{\text{trainable}}^{\text{front-end}}(\hat{x}_t^b), \tag{15}$$

where $F_{\text{trainable}}^{\text{front-end}}(\cdot)$ denotes the cloned front-end network, which is trainable during the training process alongside the rest of the base layer, and $\hat{r}_t$ is the resultant feature tensor. Then $\hat{r}_t$ is fed to the back-end part of the original pre-trained task network, $F_{\text{original}}^{\text{back-end}}(\hat{r}_t)$, where $F_{\text{original}}^{\text{back-end}}(\cdot)$ is not trained during the training process.

The primary advantage of this methodology is to eliminate the need for ground-truth labels for the base layer. The amount of labeled data for video (especially uncompressed video, which is needed in our application) is very limited, and it makes sense to use it all for testing. So instead, we train the base layer to match the features r_t of the pre-trained network at a particular layer, where

$$r_t = F_{\text{original}}^{\text{front-end}}(x_t), \tag{16}$$

and this way, we do not need to use labeled video in training.

The selection of the layer for feature matching will depend on the application constraints. The rate-distortion results in [39, 40] suggest that better performance is achieved when matching deeper layers. However, this requires making $F_{\text{trainable}}^{\text{front-end}}(\cdot)$ larger, which in turn increases the complexity of the base layer. For the experiments in this study, we chose the first five layers of YOLOv5 as the front-end to demonstrate the performance of the system. In practice, the optimal choice will depend on the computational resources, allowed latency, available bandwidth, etc., in a particular application scenario.

4.2.5 Base layer loss function

The loss function for training the proposed base layer is defined as follows:

$$L_{base}(t) = R_{base}(t) + \lambda_{base}\,\mathrm{MSE}(r_t, \hat{r}_t), \tag{17}$$

where $\mathrm{MSE}(r_t, \hat{r}_t)$ is the mean squared error between features derived from an uncompressed frame (r_t) and features decoded from the base layer ($\hat{r}_t$), λ_{base} is the Lagrange multiplier, and $R_{base}(t)$ is the total rate of the base layer for coding x_t, which is defined as:

$$R_{base}(t) = R_{base}^{\mathrm{motion}}(t) + R_{base}^{\mathrm{signal}}(t), \tag{18}$$

where $R_{base}^{\mathrm{motion}}(t)$ is the rate of the motion bitstream produced by coding v_t^b conditioned on $\bar{v}_t^b$ plus the rate associated for its hyper prior, and $R_{base}^{\mathrm{signal}}(t)$ is the rate for coding x_t conditioned on $\bar{x}_t^b$ plus the rate associated for its hyper prior. As seen from the above equations, the goal of the base-layer codec is to preserve task-relevant features while minimizing the bitrate. This means that the base layer codec is not asked to reconstruct (or code) task-irrelevant information, which usually results in a lower bitrate compared to conventional coding approaches that try to preserve the overall input fidelity.

A visual example of the base frame ($\hat{x}_t^b$) in the trained system is shown in Fig. 5, along with the original frames (x_t).$x_t - \hat{x}_t^b$. As is evident from this figure, the base frame omits certain details of the original frame, which are deemed unnecessary for the object detection task, in order to curtail the rate. Consequently, the base layer can be encoded at a significantly lower rate than the original frame.

4.3 The enhancement layer

To implement the enhancement layer of the proposed system, we used the conditional coding-based framework proposed in [14] as the basis. However, we made some modifications to fit it into our scalable codec, as will be discussed in the following subsections. The components of the enhancement layer are shown in Fig. 6. In summary, the enhancement layer works as follows:

- The current frame and the previously decoded video frame from the enhancement layer are used to estimate a motion flow map, which is subsequently encoded by a motion coder.



Fig. 5 A visual example of the base frame at 0.11 bpp. The original frame x_t (left), and the base frame $\hat{x}_t^b$ (right). $x_t - \hat{x}_t^b$)

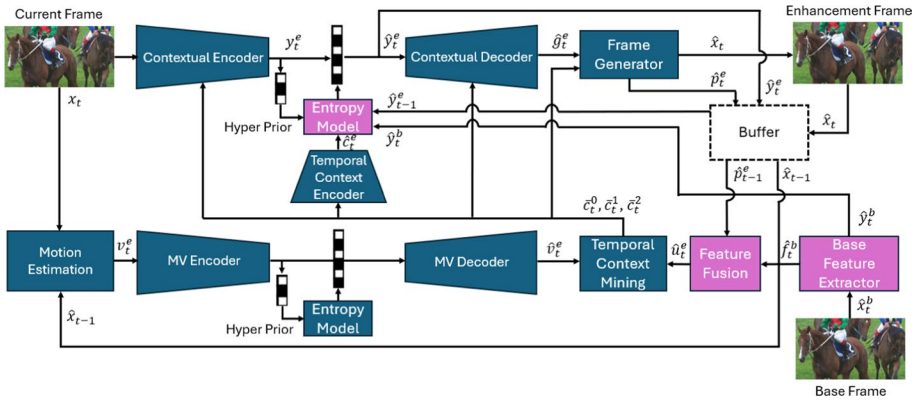


Fig. 6 The block diagram of the enhancement layer of the proposed system. It gets the current frame x_t and the base frame $\hat{x}_t^b$ as inputs and generates the enhancement frame $\hat{x}_t$ as output. Our proposed enhancement layer is built upon DCVC-TCM [14]. The purple blocks are our proposed components

- The base frame generated by the base layer is processed by a base feature extractor, and the resulting base features are fused with the latent representations of the previously reconstructed frame in the enhancement layer.
- The fused features from the previous step, combined with the decoded motion flow map, are input into a temporal context mining module to learn motion-aligned temporal contexts. These contexts capture a multi-scale representation of the current frame, each serving as a predictor for the frame within the latent space.
- The current frame is then encoded by a conditional contextual codec, which uses the generated temporal contexts as predictors. This codec employs an entropy model with parameters estimated from the temporal contexts, along with the latent representations of the base frame and the previously decoded frame in the enhancement layer.
- The contextual decoder, together with a frame generator, reconstructs the encoded video frame in the enhancement layer, which is called "Enhancement Frame".

We will now provide a detailed description of each component within the enhancement layer.

4.3.1 Motion estimator and coder

The current frame x_t and the previously reconstructed frame in the enhancement layer, $\hat{x}_{t-1}$, are fed to a motion estimation network to estimate a flow map, v_t^e, in the enhancement layer. Similar to the base layer, we also use PWC-Net [38] for motion estimation in the enhancement layer. The estimated flow map is then coded by a hyper-prior-based encoder [6], P_π, to obtain the compressed motion bitstream in the enhancement layer. During training, the hyper-prior-based encoder provides the rate estimate for the motion information in the enhancement layer, $R_{enh}^{\text{motion}}(t)$. The motion bitstream is then decoded by the corresponding hyper-prior-based decoder, P_π^{-1}, to obtain the decoded flow map, $\hat{v}_t^e$, which provides an estimated motion vector for each pixel within the current frame in the enhancement layer.

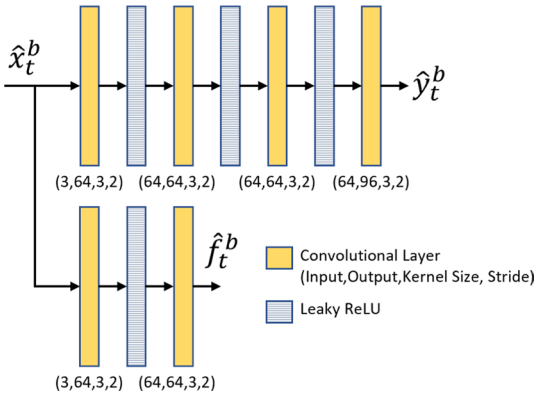


Fig. 7 The structure of the base feature extractor network

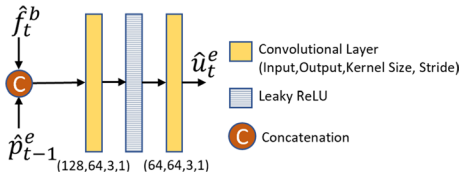


Fig. 8 The structure of the feature fusion network used in the enhancement layer

4.3.2 Base feature extractor and fusion

As shown in Fig. 7, the base frame $\hat{x}_t^b$ is first fed to a base feature extractor network to obtain two output feature tensors: $\hat{f}_t^b$ and $\hat{y}_t^b$. Note that the base feature extractor network is different from the front-end of the task model. The first feature tensor, $\hat{f}_t^b$, is then fused with the features of the previous frame reconstructed by the frame generator, $\hat{p}_{t-1}^e$, to generate a new feature tensor $\hat{u}_t^e$. The feature fusion is performed by a simple network with two convolutional layers whose structure is shown in Fig. 8. The second output of the base feature extractor, $\hat{y}_t^b$, is used by the entropy model, which is described in Sect. 4.3.7.

4.3.3 Temporal context mining

The goal of the temporal context mining (TCM) module is to generate a set of multi-scale temporal contexts as predictors for conditional coding of the current frame using the contextual encoder. The structure of TCM is shown in Fig. 9. To generate the temporal contexts to encode the current frame, $\hat{u}_t^e$ and the decoded flow map $\hat{v}_t^e$ are fed to the TCM module proposed in [14]. TCM uses $\hat{v}_t^e$ and warps $\hat{u}_t^e$ at three different scales to obtain three contexts $\{\bar{c}_t^0, \bar{c}_t^1, \bar{c}_t^2\}$, each at a different scale. These contexts provide rich latent temporal information for coding the current frame in the enhancement layer. After that, these three contexts are combined through a temporal prior network whose structure is shown in Fig. 10. The output of this network is $\hat{c}_t^e$, which conveys the temporal context of the current frame.

In [14], $\hat{p}_{t-1}^e$ is used directly as input to the TCM module. However, in our design, we fuse $\hat{p}_{t-1}^e$ with $\hat{f}_t^b$ through the feature fusion network, and the resulting output, $\hat{u}_t^e$, is used as input to the TCM module. In this way, the generated temporal contexts leverage

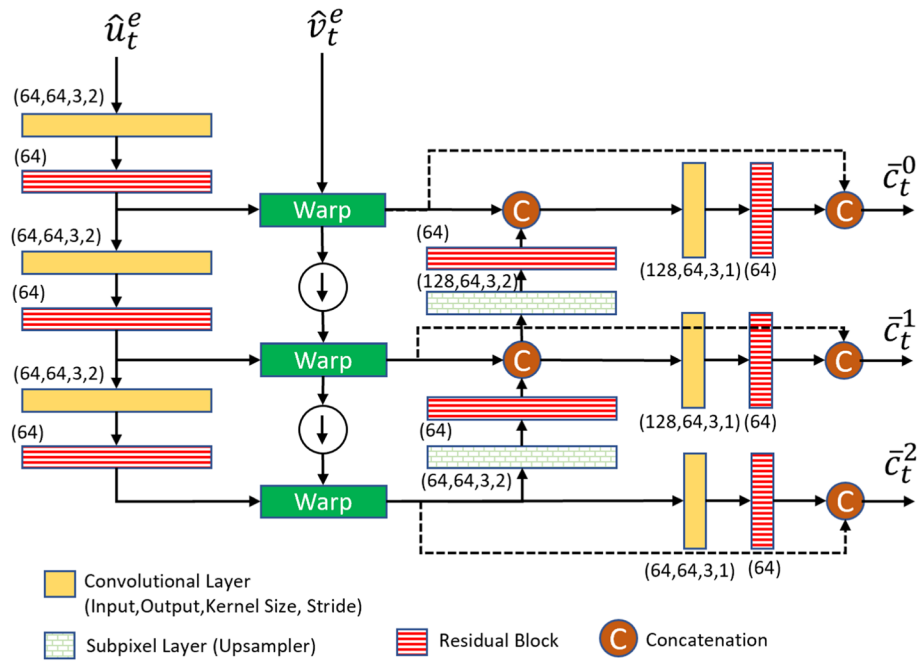


Fig. 9 The structure of the temporal context mining (TCM) network

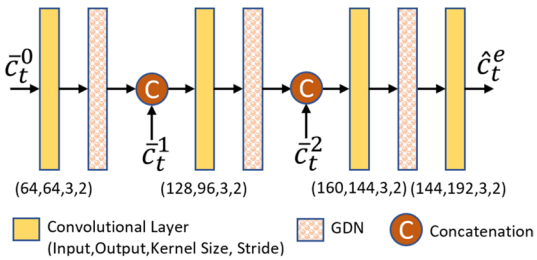


Fig. 10 The structure of the temporal prior network

the information provided in both the base frame and the previous reconstructed frame in the enhancement layer.

4.3.4 Contextual encoder

The structure of the contextual encoder is shown in Fig. 11. As shown in this figure, the current frame x_t is encoded by a conditional contextual encoder, R_π, to produce a latent representation y_t^e. The conditional contextual encoder uses the three temporal contexts $\{\bar{c}_t^0, \bar{c}_t^1, \bar{c}_t^2\}$ for conditional coding of x_t. These contexts are produced by the TCM module described in Sect. 4.3.3. The obtained latent representation y_t^e is then quantized and compressed by an arithmetic encoder whose symbol probabilities are estimated by a hyper prior-based entropy model [6]. Using this approach, a compressed contextual bitstream is produced along with a side compressed bitstream for signaling the hyper-priors h_t^e. The contextual encoder uses an entropy model, which is described in Sect. 4.3.7 below.

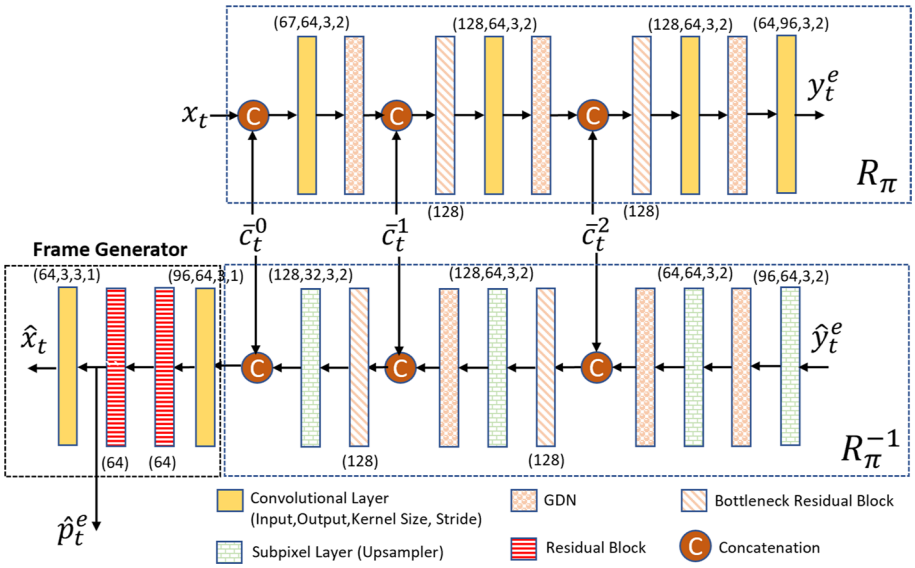


Fig. 11 The structure of the context encoder R_π, the context decoder R_π^{-1}, and the frame generator. GDN represents the generalized divisive normalization layer [41]

4.3.5 Contextual decoder

The structure of the contextual decoder is shown in Fig. 11. The contextual bitstream produced by the contextual encoder is decompressed at the decoder by an arithmetic decoder to obtain $\hat{y}_t^e$, which is then fed to a conditional contextual decoder, R_π^{-1}. Similarly, the hyper-priors of the current frame are decoded from the side bitstream to obtain the temporal hyper-prior $\hat{h}_t^e$. A copy of $\hat{y}_t^e$ is stored in the generalized decoded picture buffer (DPB) to be used for encoding the subsequent frames. R_π^{-1} uses the three temporal contexts and transforms $\hat{y}_t^e$ into a new latent representation $\hat{g}_t^e$, which is then fed to the frame generator to reconstruct the transmitted frame, as will be described next.

4.3.6 Frame generator

The frame generator passes its input tensor $\hat{g}_t^e$ through a number of convolutional layers and residual blocks as in [14], and reconstructs the input frame as $\hat{x}_t$. As in [14], the output of the layer before the last layer in the frame generator, $\hat{p}_t^e$, is also stored in the DPB along with $\hat{x}_t$. The feature tensor $\hat{p}_t^e$ is subsequently used by the TCM network to produce the temporal contexts.

4.3.7 Entropy model

Similar to [9, 14], we use the factorized entropy model for the hyper-prior coder, and the Laplace distribution to model the quantized latent representations in the enhancement layer, $\hat{y}_t^e$. However, to improve the performance of the entropy model, we fuse the hyper priors of the current frame, $\hat{h}_t^e$, with the temporal prior $\hat{c}_t^e$, decoded latents from the previous enhancement frame $\hat{y}_{t-1}^e$ and base features of the current frame $\hat{y}_t^b$ through an entropy parameter estimation network. In [14], only the hyper-prior $\hat{h}_t^e$ and the temporal prior $\hat{c}_t^e$ are fed to the entropy model. However, in our design, in

addition to $\hat{h}_t^e$ and $\hat{c}_t^e$, we also feed $\hat{y}_t^b$ and $\hat{y}_{t-1}^e$ to the entropy model to enrich its input information. Moreover, we add the attention layer so the system can learn where to take the relevant information for entropy modeling. The proposed entropy model is implemented using an entropy estimation network, which is described next.

The entropy estimation network: The structure of the entropy model is shown in Fig. 12. As shown in this figure, the hyper-prior $\hat{h}_t^e$ and the temporal prior $\hat{c}_t^e$ are first concatenated, and fed to three consecutive convolutional layers. The resulting feature tensor is then concatenated with $\hat{y}_{t-1}^e$ and $\hat{y}_t^b$, and the generated tensor is fed to two consecutive convolutional layers followed by an attention layer. The output of the attention layer is then chunked in half to produce two tensors, $\hat{\mu}_t^e$ and $\hat{\sigma}_t^e$, which are used to estimate the mean and variance of the Laplace distribution, respectively.

Sample frames from the BasketballDrill sequence (HEVC Class C) are also shown in Fig. 12: x_t is the current (to-be-coded) frame at time t, $\hat{x}_{t-1}^e$ is the previously decoded frame in the enhancement layer, and $\hat{x}_t^b$ is the base frame at time t. To gain insight into the operation of the entropy model, visualizations of the corresponding latent features are also shown. As seen in these sample visualizations, incorporating the attention layer into the proposed entropy parameter estimation network facilitates its ability to selectively focus on important features within the input data, resulting in more efficient parameter estimation. For instance, the attention mechanism weakens static areas that correspond to the floor in the first channel of $\hat{\sigma}_t^e$; since these areas are static, $\hat{\sigma}_t^e$ reduces towards zero, while it remains higher near moving objects, the basketball and players. We can also observe that some parts of the last channel of $\hat{\mu}_t^e$ are mostly obtained from the last channel of the base features $\hat{y}_t^b$ (e.g., those in the solid circle) while some other parts (e.g., those in the dashed circle) are mostly taken from the enhancement features of the previous frame, $\hat{y}_{t-1}^e$. By utilizing attention and feeding it both the base and previous enhancement features, the entropy model is able to leverage the information available in the current base frame as well as

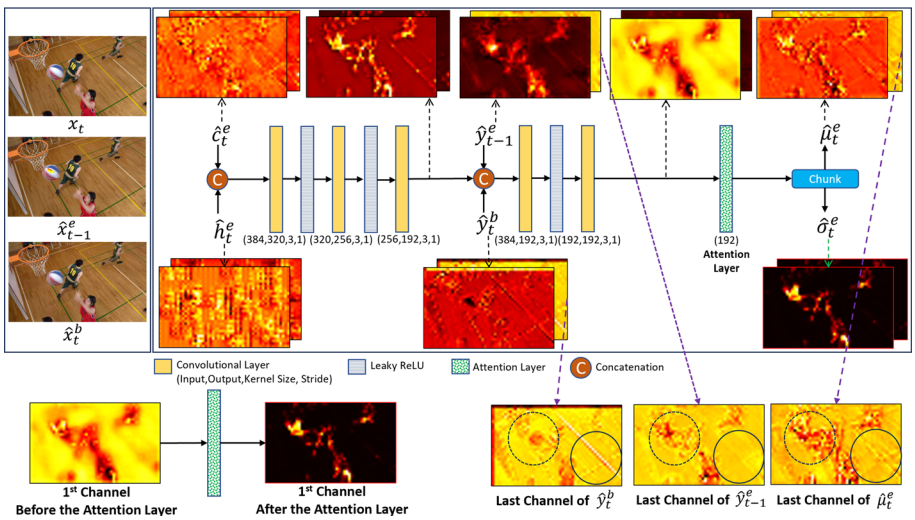


Fig. 12 The structure of the proposed entropy parameter estimation network. The latent feature maps are shown for a sample frame from BaskteballDrill (HEVC Class C)

previously reconstructed frames in the enhancement layer, thereby enhancing its ability to achieve better compression of the current enhancement frame.

Rate estimation: As will be discussed in the following section, the base and enhancement layers are trained on sequences of contiguous frames to allow the models to learn temporal relationships. Specifically, the gradients of the current frame are back-propagated to the reference frames that preceded it within a temporal window of length N frames. As a result, the tensors $\hat{y}^e_{t-1}$, $\hat{y}^b_t$ and $\hat{c}^e_t$ are endowed with substantial information not only from the current frame, but also from the preceding frames. This characteristic enables the system to take advantage of the correlations among the latent representations of multiple frames to model the entropy more effectively.

Having obtained $\hat{\mu}^e_t$, $\hat{\sigma}^e_t$, and $\theta^e_t = \{\hat{h}^e_t, \hat{c}^e_t, \hat{y}^e_{t-1}, \hat{y}^b_t\}$, the formulation for the conditional distribution of the latent representations can be obtained as follows [6]:

$$p_{\hat{y}^e_t|\theta^e_t}(\hat{y}^e_t|\theta^e_t) = \prod_i p_{\hat{y}^e_{t,i}|\theta^e_t}(\hat{y}^e_{t,i}|\theta^e_t), \tag{19}$$

where the index i represents the spatial location, and

$$\begin{aligned} p_{\hat{y}^e_{t,i}|\theta^e_t}(\hat{y}^e_{t,i}|\theta^e_t) &= \left[(\mathcal{L}(\hat{\mu}^e_{t,i}, \hat{\sigma}^e_{t,i}) * \mathcal{U}\left(-\frac{1}{2}, \frac{1}{2}\right)\right](\hat{y}^e_{t,i}) \\ &= \mathrm{cdf}\left(\hat{y}^e_{t,i} + \frac{1}{2}\right) - \mathrm{cdf}\left(\hat{y}^e_{t,i} - \frac{1}{2}\right), \end{aligned} \tag{20}$$

where $\mathcal{L}(\hat{\mu}^e_{t,i}, \hat{\sigma}^e_{t,i})$ represents the Laplacian probability density function with mean $\hat{\mu}^e_{t,i}$ and variance $\hat{\sigma}^e_{t,i}$, $*$ represents the convolution, $\mathcal{U}(-\frac{1}{2}, \frac{1}{2})$ represents the uniform probability density in the range $[-\frac{1}{2}, \frac{1}{2}]$ (which models uniform quantization), and $\mathrm{cdf}(\cdot)$ is the cumulative distribution function (CDF) of the resulting distribution. The estimated per-sample rate for coding $\hat{y}^e_t$ is given by $-\log_2(p_{\hat{y}^e_t|\theta^e_t}(\hat{y}^e_t|\theta^e_t))$, and the estimated entropy is the expectation of this quantity. This rate estimate is one of the loss terms used for training the enhancement layer, as discussed next.

4.3.8 Enhancement layer loss function

The loss function for the enhancement layer is defined as:

$$L_{enh}(t) = R_{enh}(t) + \lambda_{enh}\,\mathrm{MSE}(x_t, \hat{x}_t), \tag{21}$$

where x_t and $\hat{x}_t$ are the input and enhancement layer's decoded frame in RGB domain, respectively, at time t, λ_{enh} is the Lagrange multiplier, and $R_{enh}(t)$ is the rate estimate for coding the current enhancement-layer frame, which is defined as follows:

$$R_{enh}(t) = R^{\mathrm{motion}}_{enh}(t) + R^{\mathrm{context}}_{enh}(t), \tag{22}$$

where $R^{\mathrm{motion}}_{enh}(t)$ is the total rate estimate for coding the motion information in the enhancement layer and $R^{\mathrm{context}}_{enh}(t) = \mathbb{E}[-\log_2(p_{\hat{y}^e_t|\theta^e_t}(\hat{y}^e_t|\theta^e_t))]$.

5 Experiments

In this section, we describe the experiments performed to evaluate the performance of the proposed scalable video codec. Training details are described in Sect. 5.1, followed by the evaluation methodology in Sect. 5.2, and the experimental results for the base and enhancement layers in Sects. 5.3 and 5.4, respectively. Break-even analysis is presented in Sect. 5.5.

5.1 Training

For training the proposed system, we used the VIMEO-90K Septuplet dataset [42], which consists of 91,701 7-frame sequences with fixed resolution 448 × 256, extracted from 39K selected video clips. We randomly cropped these clips into 256 × 256 patches and used them to train the system. We used Adam [43] optimizer with batch size of 4. We employed a training strategy commonly used in recent papers [10, 14, 44], where the video codec is trained on a group of consecutive frames. In general, utilizing a larger group of frames in training can lead to better results. However, similar to [10, 14] and motivated by memory constraints, in our experiments the loss function for each layer is computed over $N = 5$ consecutive frames as follows:

$$L^*_{base} = \frac{1}{N}\sum_{t=1}^{N} L_{base}(t), \quad L^*_{enh} = \frac{1}{N}\sum_{t=1}^{N} L_{enh}(t), \tag{23}$$

where $L_{base}(t)$ and $L_{enh}(t)$ are defined as in (17) and (21), respectively. Using this training strategy, the gradients of each frame are back-propagated towards other frames in the group, which allows the system to capture temporal relationships among a group of consecutive frames.

For the base layer, we used $\lambda_{base} \in \{2, 4, 8, 16\}$. All networks in the base layer were first initialized with their pre-trained weights from [10]. For YOLOv5, we used the PyTorch implementation provided in [45], and used the checkpoint for the "small" model for the initialization. The learning rate was set to a small value 10^{-6}, and the entire layer was trained end-to-end for 10 epochs. We first trained the base layer, and then kept it frozen during the training of the enhancement layer.

For the enhancement layer, we used $\lambda_{enh} \in \{256, 512, 1024, 2048\}$. The learning rate was set to 10^{-4} for the first 5 epochs, and then reduced to 10^{-5} for the next 10 epochs. We first trained the enhancement layer with $\lambda_{enh} = 2048$ (highest rate), then all lower-rate models were initialized from this model.

5.2 Evaluation methodology

5.2.1 Evaluation datasets

To evaluate a video codec intended for both object detection and input frame recovery, it is necessary to test it on datasets consisting of raw (uncompressed) video and associated object labels. To our knowledge, only two such datasets are currently available—SFU-HW-Objects-v1 [46] and TVD [47]—and both are being used in the MPEG-VCM standardization effort [17]. The SFU-HW-Objects-v1 dataset is based on raw YUV420 video sequences that were used in the development of video coding standards such as HEVC

[4] and VVC [5]. These sequences are usually referred to as "HEVC test sequences" and come in several classes. We used sequences from three classes: Class B (1920×1080 resolution), Class C (832×480), and Class D (416×240). Each sequence contains between 240–600 frames and their frame rates range from 24 to 50 frames per second. A subset of these sequences was labeled with COCO7-style object labels. The TVD dataset includes 86 sequences, each consisting of 65 frames. However, object labels are only provided for 55 unique video triplets (3-frame sequences).

5.2.2 Benchmark methods

We used the following video codecs as benchmarks: HEVC, VVC, DCVC-TCM [14], CANF-VC [10], and DCVC [9]. For HEVC, we used HM 18.0 [48] with LDP profile (encoder_lowdelay_P_main.cfg), and for VVC, we employed VTM 23.4 [49] with LDP profile (encoder_lowdelay_P_vtm.cfg). Among these, VTM was chosen as the anchor against which all other codecs are compared. For HM, which is the standard implementation of HEVC [4], the quantization parameter (QP) values were {27, 29, 32, 38}. For VTM, which is the standard implementation of VVC [5], QP values were {22, 24, 27, 29}. The internal color space for HM and VTM was YUV420 (8 bit). Also, four past reference frames were used for motion estimation in these conventional codecs. However, all the tested learned codecs (CANF-VC, DCVC-TCM, LCCM-VC, DCVC, and the proposed method) use only one reference frame in their optical flow estimators. As reported in [14], if we use only one reference frame for the conventional codecs instead of four, their performance will be degraded considerably. In line with the common practice in previous learned video codecs [10, 14], in the experiments, we set the IntraPeriod (also known as Group Of Pictures, GOP) to 32 for all codecs when testing them on SFU-HW-Objects-v1 to be able to compare our results with other existing works. For intra-frame coding of the first frame in each GOP, we used the intra-frame coder from CANF-VC for the base layer and the intra-frame coder from DCVC-TCM for the enhancement layer. On the TVD dataset, the GOP size is set to 3 because sequences are 3-frames long. For CANF-VC, DCVC-TCM, LCCM-VC, and DCVC, which require RGB frames, we first converted all YUV 420 videos to RGB format using ffmpeg [50], following the ITU-R BT.709 recommendation [51]. The resulting RGB frames were then used for both training and inference.

5.2.3 Evaluation metrics

Following the existing practice in the learned video coding literature [9, 10, 14, 52], to evaluate the performance of various codecs, the bitrates were measured in bits per pixel (bpp) and the frame reconstruction quality was measured by RGB peak signal-to-noise ratio (PSNR-RGB) and RGB multi-scale structural similarity index metric (MS-SSIM-RGB) [53]. Object detection accuracy was measured by mean average precision (mAP) [33]. The mAP is a popular metric used to evaluate the accuracy of object detection models. It is computed by first calculating the average precision (AP) for each class, which is the area under the precision–recall curve. Precision is the ratio of correctly predicted positive samples to all predicted positives, while recall is the ratio of correctly predicted positives to all actual positives. The mAP is then obtained by averaging the APs across all classes, giving an overall performance score for the model's ability to detect objects

across different categories. Following the methodology used in MPEG-VCM [17], these quantities can be agglomerated into a single numerical score called Bjøntegaard Delta Rate (BD-Rate) [54], which computes the average bitrate difference between two codecs at the equivalent quality level, where the quality may be measured by PSNR-RGB, MS-SSIM-RGB, or mAP, as the case may be. So, for example, if a certain codec achieves −10% BD-Rate-mAP relative to a given anchor, it means that it achieves, on average, 10% bitrate reduction relative to the anchor while producing the same mAP.

5.3 Object detection performance

First, we evaluate the object detection performance. For the proposed codec, we only need to decode the base-layer bitstream and feed it to the pre-trained YOLOv5 back-end. For other codecs, we decode each frame and then feed it to the pre-trained YOLOv5. Figure 13 shows the mAP (%) vs. bitrate (bpp) on the three classes of sequences from the SFU-HW-Objects-v1 dataset. The straight horizontal line in each graph is the performance of YOLOv5 without input frame compression. The legend in each graph shows BD-Rate-mAP percentage in the brackets, relative to the VTM anchor. Negative values indicate bit savings relative to the anchor at the equivalent mAP. BD-Rate-mAP results are also summarized in Table 1, where the best result in each row is highlighted in bold. The table also includes the results on the TVD dataset. However, since the sequences in the TVD dataset are short (3 frames each), the performance of most codecs is similar. The TVD sequences seem too short to allow distinguishing the various codecs' ability to capture long-range relationships among features relevant to object detection. The last two rows in Table 1 show the average BD-Rate-mAP results, averaged according to the number of frames in each dataset. We see that the overall performance (bottom row) is dominated by the performance on SFU-HW-Objects-v1, since this dataset has many more frames than TVD. Overall, the results indicate that the proposed codec in the base layer outperforms all other codecs. HM comes as the second best, followed by LCCM-VC and DCVC-TCM.

Figure 14 provides a visual example of object detection on a sample frame from the PartyScene sequence (HEVC Class C). In this figure, we show the bounding box of the detected objects overlaid on the reconstructed frame. For the proposed method, we show the base frame because only base-layer information is used. As seen in this example, the proposed method is capable of achieving better detection accuracy at a lower rate compared to other methods. Specifically, we note that in the frames produced by

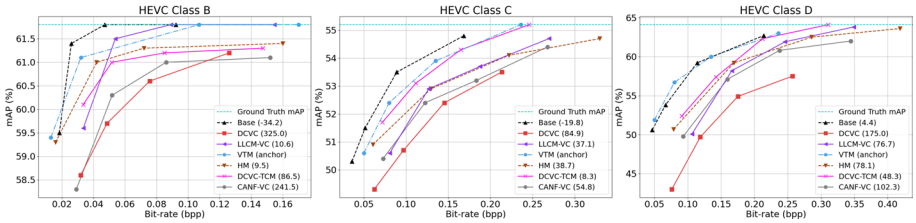


Fig. 13 Comparing the performance of various methods in terms of object detection mAP (%) on three datasets. The straight horizontal line in each graph is the performance of YOLOv5 without input compression. In the legend of each graph, the BD-Rate(%) of the corresponding method is also shown with respect to VTM (anchor)

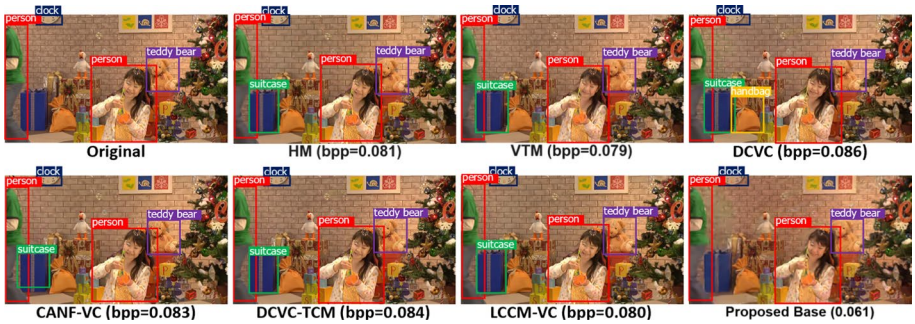


Fig. 14 Visualizing object detection performance of various methods on a sample frame from PartyScene (HEVC Class C). The ground-truth object labels come from the SFU-HW-Objects-v1 dataset

Table 1 Object detection performance in terms of BD-Rate-mAP (%) on different datasets

Dataset	CANF-VC [10]	DCVC-TCM [14]	LCCM-VC [23]	DCVC [9]	Base	HM
SFU-HW-Objects-v1 (HEVC Class B)	241.5	86.5	10.6	325	**−34.2**	9.5
SFU-HW-Objects-v1 (HEVC Class C)	54.8	8.3	37.1	84.9	**−19.8**	38.7
SFU-HW-Objects-v1 (HEVC Class D)	102.3	48.3	76.7	175	**4.4**	78.1
TVD	0.1	0.2	0.1	33.1	**−0.5**	0.2
Average (SFU-HW-Objects-v1)	136.2	48.8	40.5	198.9	**−17.1**	41.1
Average	134.2	48.18	39.9	196.5	**−16.8**	43.5

The anchor is VTM. "Base" represents the proposed method in the base layer

Table 2 Frame reconstruction performance in terms of BD-Rate-PSNR (%) on different datasets

Dataset	CANF-VC [10]	DCVC-TCM [14]	LCCM-VC [23]	DCVC [9]	Base	Enh	Base+Enh	HM
HEVC Class B	59.8	24.1	55.5	101.7	162.4	29.1	65.3	**23.4**
HEVC Class C	56.1	36.9	52.0	99.4	136.5	32.5	68.5	**11.6**
HEVC Class D	58.1	30.4	53.6	30.4	133.7	26.7	62.4	**17.4**
TVD	27.5	31.4	25.6	67.1	83.4	19.6	44.5	**9.6**
Average (HEVC)	58.0	30.3	53.7	77.9	144.7	29.4	65.4	**17.6**
Average	57.6	30.3	53.3	77.7	143.9	29.3	65.1	**17.5**

The anchor is VTM. The last three columns from the right show the results of the three cases of the proposed method

other codecs, the gift box is often mis-detected as a suitcase, and the gift bag is confused with a handbag. While these objects might be similar to some extent, the compression artifacts introduced by other codecs cause the detector to produce incorrect detections, whereas our method preserves the relevant features well, avoiding mis-detection even at lower rates than other methods.

5.4 Frame reconstruction performance

When it comes to the performance of reconstructing compressed frames for human viewing, the situation is straightforward for conventional video codecs, whether learned or handcrafted: we simply decode the compressed bitstream, generate the decoded frames, and measure their quality using a metric (PSNR-RGB or MS-SSIM-RGB in this work). However, for the proposed codec, the following options can be considered:

Table 3 Frame reconstruction performance in terms of BD-Rate-MS-SSIM (%) on different datasets

Dataset	CANF-VC [10]	DCVC-TCM [14]	LCCM-VC [23]	DCVC [9]	Base	Enh	Base+Enh	HM
HEVC Class B	56.7	1.2	39.1	72.3	60.8	**3.6**	34.4	9.2
HEVC Class C	51.8	6.7	44.1	59.5	51.1	**4.6**	35.5	37.9
HEVC Class D	70.1	11.2	61.7	56.3	53.4	**0.3**	33.5	41.0
TVD	15.1	8.2	13.3	9.6	39.3	**0.7**	16.2	14.6
Average (HEVC)	59.4	6.2	48.0	63.0	55.3	**2.8**	34.4	28.7
Average	58.8	6.2	47.5	62.2	55.0	**2.8**	34.1	28.5

The anchor is VTM. The last three columns from the right show the results of the three cases of the proposed method

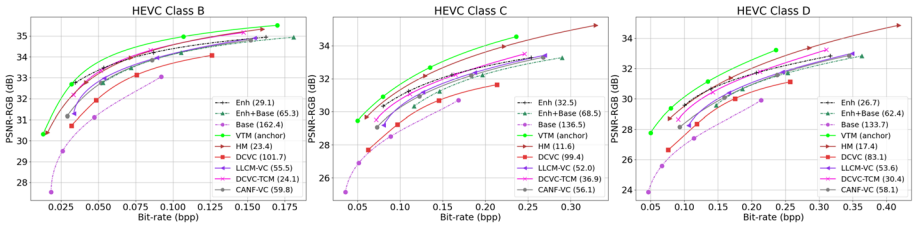


Fig. 15 Comparing the input reconstruction performance of various methods in terms of PSNR-RGB (dB) on three datasets. In the legend of each graph, the BD-Rate(%) of the corresponding method is also shown with respect to VTM (anchor). "Base", "Enh", and "Base+Enh" represent different cases of the proposed system

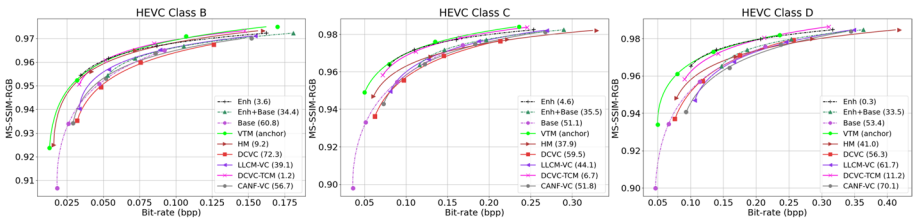


Fig. 16 Comparing the input reconstruction performance of various methods in terms of MS-SSIM-RGB on three datasets. In the legend of each graph, the BD-Rate(%) of the corresponding method is also shown with respect to VTM (anchor). "Base", "Enh", and "Base+Enh" represent different cases of the proposed system

- **Option 1:** First, note that the base layer of the proposed system generates video frames and stores them in the base frame buffer, as shown in Fig. 3. Such frames are intended for object detection rather than human viewing; they preserve enough information to detect objects at low rates, but lack some of the information that is irrelevant for object detection. Nevertheless, we could ask what their quality is in terms of frame reconstruction metrics. In Figs. 15-16 and Tables 2-3, we refer to this approach as "**Base**". In this case, only the base-layer bitrate is counted, since base frames are generated from the base-layer bitstream only.
- **Option 2:** The intended use of the proposed codec is in a closed-loop automated analytics system, where analysis is performed using object detection (base layer) information, and frame reconstruction is initiated depending on the results of the base-layer analysis. For example, in a traffic monitoring system, object detection runs continuously. When the arrangement of detected objects suggests a traffic violation, the enhancement layer is turned on to provide additional information for frame reconstruction and human viewing. Hence, given that base-layer informa-

tion is already available at the analysis server, only the enhancement layer bitstream needs to be communicated to provide high-quality frame reconstruction. The bitrate in this case is the enhancement-layer bitrate, and the frame quality is measured on the enhancement-layer frames. This approach is termed "**Enh**" in Figs. 15-16 and Tables 2-3. It is worth noting that the MPEG-VCM evaluation framework [55] suggests comparing this approach against conventional coding approaches. The goal of MPEG-VCM is stated as [55]: "*The bitrate of the additional compressed bitstream shall be less than the bitrate of the bitstream at similar quality as measured by PSNR, which is the output of the VVC encoding of the unprocessed video.*"

- **Option 3:** Finally, we can also count the total bitrate (base + enhancement layer) for the proposed codec, along with enhancement-layer frame quality. This is referred to as "**Base+Enh**" in Figs. 15-16 and Tables 2-3. It is debatable whether the comparison of conventional coding against this case is realistic. Such a comparison may make sense if we knew *in advance* that a situation of interest (e.g., traffic violation in our earlier example) would occur at some point in time, without performing any visual analysis. In such a case, our codec would still have to encode both base and enhancement layer bitstreams, even though the result of the analysis is apparently known ahead of time. Nevertheless, we included these "oracle-style" results for completeness, even though they show less favorable performance of our codec.

We now evaluate the proposed method using these three options and compare its performance with other benchmark methods through both quantitative and qualitative analyses.

5.4.1 Quantitative results using PSNR-RGB

Figure 15 shows the PSNR-RGB (in dB) versus bitrate for various cases on the three classes of HEVC video sequences. The legend shows BD-Rate-PSNR (%) relative to VTM in the brackets. These results are also tabulated in Table 2, along with BD-Rate-PSNR on the TVD dataset. As seen from these results, for the input frame reconstruction task, the best-performing method is VTM (as all values in this table are positive, meaning that all methods performs worse than VTM) followed by HM and then our proposed Enh. The gap between VTM and our Enh is about 29%. This is not surprising, given that conventional video codecs are known to perform very well on the PSNR metric. It is interesting to note that even though our Base method uses the base layer bitstream only (seemingly a low bitrate), it performs the worst among all the methods tested in Table 2. This is because BD-Rate-PSNR between a given codec and the anchor is measured at the equivalent PSNR, and our Base method needs a lot of bits to reach the equivalent PSNR to VTM. Our base layer was designed to support object detection, so it lacks the details needed to achieve high PSNR, and therefore needs to use a lot of bits to bring those details back in.

5.4.2 Quantitative results using MS-SSIM-RGB

Figure 16 and Table 3 show analogous results but with MS-SSIM-RGB as the frame quality metric. Learned codecs are known to perform better on MS-SSIM-RGB than on PSNR-RGB, even when MSE is used in their loss function (21). Indeed, this is seen in

Fig. 16 and Table 3. Specifically, although VTM still comes out as the best codec on BD-Rate-SSIM, the gap between it and our Enh (the second-best method) is now about 2.8%. DCVC-TCM (the third-best method) also comes to within a few percentage points of VTM. Almost all other codecs perform comparatively better in Table 3 than in Table 2. It is interesting to note that our Base method records the highest jump in performance between Table 2 and Table 3. This suggests that object-relevant information, which is preserved in our base layer, is also favorable to quality metrics such as MS-SSIM-RGB.

5.4.3 Qualitative results

Figure 17 shows a patch from a sample frame from RaceHorses (HEVC Class C) reconstructed by various codecs at a similar bitrate: 0.42–0.44 bpp for all codecs, except our Enh at 0.37 bpp. As seen from these examples, learned codecs tend to suppress some details, especially in the grass in the background; among the learned codecs, LCCM-VC and the proposed one seem to have recovered most of these details. Figure 18 shows visual comparisons of base and enhancement-layer (reconstructed) frames produced by our codec. It is evident that base frames lack detail, but they preserve enough detail needed to identify objects, which is what the base layer is designed for.

5.5 Break-even analysis

In the preceding sections, we provided experimental results for the machine task (object detection) and human viewing separately. We saw that the proposed scalable codec performs better than the alternatives on the machine task (Table 1) while VTM is the best-performing codec on frame reconstruction for human viewing (Tables 2 and 3).

So, which codec is better—VTM or the proposed one? That depends on how frequently one expects human viewing will be required in a particular application: whenever human viewing is needed, VTM is better, but when the system performs the machine task only, the proposed codec is better. To enable numerical comparison of



Fig. 17 Comparing the input reconstruction performance of various methods on a patch from a sample frame of RaceHorses (HEVC Class C). The (bpp, PSNR-RGB, MS-SSIM-RGB) for HM, VTM, DCVC, CANF-VC, DCVC-TCM, LCCM-VC, the proposed method (Base+Enh), and the proposed method (Enh) is (0.43,33.64,0.9759), (0.43,33.96,0.9807), (0.44,32.65,0.9678), (0.42,33.50,0.9750), (0.43,33.61,0.9760), (0.44,33.78,0.9771), (0.42,33.60,0.9758), and (0.37,33.60,0.9758), respectively. Note that the proposed Enh and the proposed Base+Enh produce the same frame—indicated by "Proposed" in the figure—only the bits are counted differently



Fig. 18 Three examples of the visual quality of the base frame and the corresponding enhancement-layer reconstructed frame. Top row: a frame from RaceHorses (HEVC Class D, base rate=0.13, enhancement rate=0.05); middle row: a frame from BasketballPass (HEVC Class D, base rate=0.07, enhancement rate=0.06); bottom row: a frame from the TVD dataset (base rate=0.16, enhancement rate=0.08)

Table 4 The break-even point ϕ against VTM

Frame quality metric	Enh	Base+Enh
PSNR-RGB	0.37	0.21
MS-SSIM-RGB	0.86	0.33

codecs in such a scenario, we adopt *break-even analysis* from [28]. The gist of break-even analysis is as follows. From Table 1, the proposed codec is, on average, 16.8% more efficient than VTM when the machine task is used. In other words, it uses 0.832 (or 83.2%) of the bits used by VTM in this case. Meanwhile, from Table 2, the proposed codec in the Enh regime is, on average, 29.3% less efficient than VTM (it uses 1.293 of the VTM bits) on frame reconstruction for human viewing, when frame quality is measured by PSNR-RGB. Let $\phi \in [0, 1]$ be the fraction of time that input frame reconstruction is needed for human viewing. The amount of bits used by our system will be less than or equal to that used by VTM if

$$(1 - \phi) \cdot 0.832 + \phi \cdot 1.293 \leqslant 1. \tag{24}$$

Solving for ϕ that achieves equality in (24), we obtain the break-even point of $\phi = 0.37$. That is to say, if input reconstruction for human viewing is needed less than 37% of the time, our system will provide overall bit savings over VTM. However, if MS-SSIM-RGB is used as the frame quality metric, we obtain $\phi = 0.86$. Using this analysis, the proposed method in the Enh regime always outperforms ($\phi = 1$) the other codecs in this study, meaning that it provides bit savings regardless of the fraction of time used for human viewing. For completeness, Table 4 also shows the break-even points of Base+Enh against VTM. Although Base+Enh results are less favorable to our method and less

realistic in a practical setting, as discussed earlier, even in this case our method provides gains over VTM when human viewing is needed less than 21% of the time.

But how often is human viewing needed in these kinds of applications-traffic monitoring, surveillance, etc.? We do not have a precise answer to this question. One reference point could be a report published in 2015 [56], suggesting that less than 1% of all surveillance video is watched live. Given the proliferation of surveillance technology and likely much slower increase (if any) of human viewers of surveillance video, the percentage is almost certainly lower now. Although "human viewing" in our system is not the same as "live viewing", the figure of 1% ($\phi = 0.01$) can serve as a ballpark estimate, and we see in Table 4 that the break-even points of our system are many-fold higher than that. In other words, our system could comfortably provide gains over VTM in a practical surveillance scenario.

5.6 Ablation study

In this section, we provide the results of an ablation study to demonstrate the efficiency of our proposed system. In our ablation study, we considered the following cases:

- **Case 1**: Reusing the motion flow map of the base layer for the enhancement layer: In our proposed system, motion estimation is performed independently for both the base and enhancement layers, with the resulting motion flow maps encoded and transmitted separately to the decoder. One potential optimization is to share the base layer's motion flow map with the enhancement layer, eliminating the need for a separate motion bitstream in the enhancement layer. We tested this approach on HEVC Class B. However, the results showed that using the base layer's flow map for the enhancement layer degraded the BD-Rate-PSNR performance by approximately 1%. Therefore, we opted to maintain separate motion estimation and coding networks for each layer.
- **Case 2**: Freezing the cloned front-end network in the base layer: As discussed in Sect. 4.2.4, during the training of the base layer, the cloned front-end network, $F^{\text{front-end}}_{\text{trainable}}(.)$, is allowed to be trained. A possible consideration is whether freezing this network might yield better results. We tested this approach on HEVC Class B, and the results showed that training this network led to approximately 2% better rate savings. Therefore, in our proposed system, we did not freeze the weights of this network during training.
- **Case 3**: The role of the proposed entropy model: As discussed in Sect. 4.3.7, our proposed entropy estimation module enhances the performance of the entropy model proposed in [14] by fusing the hyper priors of the current frame, $\hat{h}^e_t$, with the temporal prior $\hat{c}^e_t$, the decoded latents from the previous enhancement frame $\hat{y}^e_{t-1}$, and the base features of the current frame $\hat{y}^b_t$ through an entropy parameter estimation network. In contrast, the approach in [14] feeds only the hyper-prior $\hat{h}^e_t$ and the temporal prior $\hat{c}^e_t$ to the entropy model. Our design, however, enriches the input information by including $\hat{y}^b_t$ and $\hat{y}^e_{t-1}$ in the entropy model. Additionally, we incorporate an attention layer to enable the system to learn where to focus for effective entropy modeling. To evaluate the effectiveness of our proposed entropy estimation model, we replaced it with the original entropy model proposed in

[14] and tested the performance of the enhancement layer on HEVC Class B. The results showed that our proposed entropy model yields approximately 1.5% better rate savings in the enhancement layer.

6 Conclusion

In this manuscript, we provided a theoretical justification for conditional lossy coding, a technique whose rate-distortion bounds are better than those of the commonly used residual coding. We then presented the first end-to-end learned scalable video codec that employs conditional coding and is capable of efficiently supporting both human and machine vision. Specifically, object detection is supported by the codec's base layer, while input reconstruction for human viewing is supported by both the base and enhancement layer. We tested the proposed codec on the video object detection datasets used in MPEG-VCM, and compared it against recent learned and conventional video codecs. The results show that the proposed codec outperforms existing video codecs on the base task (object detection) while providing comparable performance to the best codecs on input frame reconstruction. We also performed break-even analysis against VTM, the best-performing codec on input frame reconstruction, which showed that in a practical scenario, our codec is able to provide bit savings against VTM if human viewing is needed less than 21% of the time.

Abbreviations

DNN:	Deep Neural Network
HEVC:	High efficiency video coding
VVC:	Versatile video coding
VCM:	Video coding for machines
CV:	Computer vision
CANF-VC:	Conditional augmented normalizing flows for video coding
LCCM-VC:	Learned conditional coding modes for video coding
TCM:	Temporal context mining

Acknowledgements
Not applicable

Author Contributions
Both authors had equal contributions.

Funding
This work was supported in part by NSERC grants RGPIN-2021-02485 and RGPAS-2021-00038.

Availability of data and materials
In our experiments, we used public HEVC standard video sequences from the following link: https://media.xiph.org/video/derf/ftp://ftp.kw.bbc.co.uk/hevc/hm-10.0-anchors/bitstreams/ra_main/ We also used the TVD dataset which is available at [47] and the SFU-HW-Objects-v1 dataset from [46].

Declarations

Competing interests
The second author (Ivan V. Bajić) is a guest editor for the following collection of JIVP: Visual coding for humans and machines

Received: 3 March 2024 Accepted: 3 November 2024
Published online: 14 November 2024

References

1. L. Duan, J. Liu, W. Yang, T. Huang, W. Gao, Video coding for machines: a paradigm of collaborative compression and intelligent analytics. IEEE Trans. Image Process. **29**, 8680–8695 (2020)
2. MPEG-CDVS: Compact descriptors for visual search. (2015). ISO/IEC JTC 1 15938-13
3. MPEG-CDVA: Compact descriptors for video analysis. (2019). ISO/IEC JTC 1 15938-15
4. G.J. Sullivan, J.R. Ohm, W.J. Han, T. Wiegand, Overview of the high efficiency video coding (HEVC) standard. IEEE Trans. Circu. Syst. Video Technol. **22**(12), 1649–1668 (2012)
5. B. Bross, Y.K. Wang, Y. Ye, S. Liu, J. Chen, G.J. Sullivan, J.R. Ohm, Overview of the versatile video coding (VVC) standard and its applications. IEEE Trans. Circu. Syst. Video Technol. **31**(10), 3736–3764 (2021)
6. J. Ballé, D. Minnen, S. Singh, S.J. Hwang, N. Johnston. Variational image compression with a scale hyperprior. In: Intl. Conf. on Learning Representations (ICLR), pp. 1– 23 (2018)
7. D. Minnen, J. Ballé, G.D. Toderici, Joint autoregressive and hierarchical priors for learned image compression. In: Advances in Neural Information Processing Systems, vol. 31 (2018)
8. Z. Cheng, H. Sun, M. Takeuchi, J. Katto, Learned image compression with discretized gaussian mixture likelihoods and attention modules. In: Proc. IEEE/CVF CVPR (2020)
9. J. Li, B. Li, Y. Lu, Deep contextual video compression. In: Advances in Neural Information Processing Systems (2021)
10. Y.H. Ho, C.P. Chang, P.Y. Chen, A. Gnutti, W.H. Peng, CANF-VC: Conditional augmented normalizing flows for video compression. In: Proc. ECCV (2022)
11. Y.H. Ho, C.C. Chan, W.H. Peng, H.M. Hang, M. Domanski, ANFIC: image compression using augmented normalizing flows. IEEE Open J. Circu. Syst. **2**, 613–626 (2021)
12. T. Ladune, P. Philippe, W. Hamidouche, L. Zhang, O. Deforges, Conditional coding for flexible learned video compression. In: Neural Compression From Information Theory to Applications–Workshop@ ICLR 2021 (2021)
13. E. Agustsson, D. Minnen, N. Johnston, N., J., B., J., H.S., F. Toderici, Scale-space flow for end-to-end optimized video compression. In: Proc. IEEE/CVF CVPR, pp. 8503– 8512 (2020)
14. X. Sheng, J. Li, B. Li, L. Li, D. Liu, Y. Lu, Temporal context mining for learned video compression. IEEE Trans. Multimed. **25**, 7311–7322 (2023)
15. W. Zhang, T. Yao, S. Zhu, A.E. Saddik, Deep learning-based multimedia analytics: a review. ACM TOMCCAP **15**, 1–26 (2019)
16. J. Ascenso, E. Alshina, T. Ebrahimi, The JPEG AI standard: providing efficient human and machine visual data consumption. IEEE MultiMedia **30**(1), 100–111 (2023)
17. M. Rafie, L. Yu, Y. Zhang, S. Liu, [VCM] call for evidence for video coding for machines. (2020). ISO/IEC JTC 1/SC 29/WG 2, m55605
18. Y. Hu, S. Yang, W. Yang, L.-Y. Duan, J. Liu, Towards coding for human and machine vision: a scalable image coding approach. In: Proc. IEEE ICME (2020)
19. K. Liu, D. Liu, L. Li, N. Yan, H. Li, Semantics-to-signal scalable image compression with learned revertible representations. Int. J. Comput. Vis. **129**, 1–17 (2021)
20. H. Choi, I.V. Bajić, Scalable image coding for humans and machines. IEEE Trans. Image Process. **31**, 2739–2754 (2022)
21. T.M. Cover, J.A. Thomas, *Elements of Information Theory*, 2nd edn. (Wiley-Interscience, Hoboken, NJ, 2006)
22. G. Wang, C. Wu, X. Zhang, Z. Liu, C. Shen, YOLOv5: An effective single-stage object detector. In: IEEE/CVF CVPR, pp. 14138– 14147 (2021)
23. H. Hadizadeh, I.V. Bajić, LCCM-VC: learned conditional coding modes for video coding. In: Proc. IEEE ICASSP Workshops (2023)
24. G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, Z. Gao, DVC: an end-to-end deep video compression framework. In: Proc. IEEE/CVF CVPR, pp. 11006– 11015 (2019)
25. H. Liu, M. Lu, Z. Ma, F. Wang, Z. Xie, X. Cao, Y. Wang, *Neural video coding using multiscale motion compensation and spatiotemporal context model* (IEEE Trans, Circuits and Systems for Video Technology, 2020)
26. Z. Hu, G. Lu, D. Xu, FVC: a new framework towards deep video compression in feature space. In: Proc. IEEE/CVF CVPR, pp. 1502– 1511 (2021)
27. T. Ladune, P. Philippe, W. Hamidouche, L. Zhang, O. Déforges, Optical flow and mode selection for learning-based video coding. In: IEEE 22nd International Workshop on Multimedia Signal Processing, (2020)
28. H. Choi, I.V. Bajić, Scalable video coding for humans and machines. In: IEEE Int. Workshop on Multimedia Signal Process. (MMSP) (2022)
29. H. Choi, I.V. Bajić, Latent-space scalability for multi-task collaborative intelligence. In: Proc. IEEE ICIP'21, pp. 3562–3566 (2021)
30. W. Jiang, H. Choi, F. Racapé, Adaptive human-centric video compression for humans and machines. In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) (2023)
31. Z. Huang, C. Jia, S. Wang, S. Ma, *HMFVC: a human-machine friendly video compression scheme* (IEEE Trans. Circuits Syst, Video Technol, 2022)
32. H. Lin, B. Chen, Z. Zhang, J. Lin, X. Wang, T. Zhao, DeepSVC: deep scalable video coding for both machine and human vision. In: MM '23: Proc. of the 31st ACM International Conference on Multimedia, pp. 9205– 9214 (2023)
33. J. Redmon, A. Farhadi, YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018)
34. T. Gong, K. Chen, X. Wang, Q. Chu, F. Zhu, D. Lin, N. Yu, H. Feng, Temporal ROI align for video object recognition. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), vol. 35, pp. 1442– 1450 (2021)
35. R.M. Gray, A new class of lower bounds to information rates of stationary sources via conditional rate-distortion functions. IEEE Trans. Inf. Theory **19**(4), 480–489 (1973)
36. W. Liu, A.G.B. Tsang, M. Yang, H. Wu, Y. Gong, SSD: Single shot multibox detector. In: European Conference on Computer Vision (ECCV), pp. 21– 37. Springer, Amsterdam, Netherlands (2016)
37. Y. Foroutan, A. Harell, A. Andrade, I.V. Bajić, Base layer efficiency in scalable human-machine coding. In: Proc. IEEE ICIP, pp. 3299– 3303 (2023)
38. D. Sun, X. Yang, M.Y. Liu, J. Kautz, PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In: Proc. IEEE/CVF CVPR, pp. 8934– 8943 (2018)

39. A. Harell, A. De Andrade, I.V. Bajić, Rate-distortion in image coding for machines. In: Proc. Picture Coding Symposium (PCS), pp. 199– 203 (2022)
40. A. Harell, Y. Foroutan, N. Ahuja, P. Datta, B. Kanzariya, V.S. Somayaulu, O. Tickoo, A. Andrade, I.V. Bajić, Rate-distortion theory in coding for machines and its application. arXiv preprint arXiv:2305.17295 (2023)
41. J. Ballé, V. Laparra, E.P. Simoncelli, End-to-end optimized image compression. In: ICLR 2017 (2017)
42. T. Xue, B. Chen, J. Wu, D. Wei, W.T. Freeman, Video enhancement with task-oriented flow. Int. J. Comput. Vision **127**(8), 1106–1125 (2019)
43. D.P. Kingma, J. Ba, Adam: a method for stochastic optimization. In: International Conference for Learning Representations (2015)
44. G. Lu, C. Cai, X. Zhang, L. Chen, W. Ouyang, D. Xu, Z. Gao, Content adaptive and error propagation aware deep video compression. In: Proc. ECCV, pp. 456– 472 (2020)
45. Ultralytics: The PyTorch implementation of YOLOv5. https://pytorch.org/hub/ultralytics_yolov5/ (2022-12-01)
46. H. Choi, E. Hosseini, S.R. Alvar, R.A. Cohen, I.V. Bajić, A dataset of labelled objects on raw video sequences. Data in Brief **34**, 106701 (2021)
47. X. Xu, S. Liu, Z. Li, Tencent video dataset (TVD): A video dataset for learning-based visual data compression and analysis. arXiv preprint arXiv: 2105.05961 (2021)
48. HM: Reference software for HEVC. https://vcgit.hhi.fraunhofer.de/Zhu/HM/blob/HM-16.22/cfg/encoder_lowdelay_P_main.cfg (2022-03-10)
49. VTM: Reference software for VVC. https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM (2022-03-10)
50. FFmpeg. https://ffmpeg.org/. Version 6.0, accessed September 14, 2024 (2023)
51. ITU: Recommendation ITU-R BT.709: Parameter values for the HDTV standards for production and international programme exchange. Technical Report BT.709-6, International Telecommunication Union (ITU) (2015). https://www.itu.int/rec/R-REC-BT.709
52. J. Lin, D. Liu, H. Li, F. Wu, M-LVC: multiple frames prediction for learned video compression. In: Proc. IEEE/CVF CVPR, pp. 3546– 3554 (2020)
53. Z. Wang, E..P. Simoncelli, A..C. Bovik, Multiscale structural similarity for image quality assessment. Proc. Asilomar Conf. Sign. Syst. Comput. **2**, 1398–1402 (2003)
54. G. Bjøntegaard, Calculation of average PSNR differences between RD-curves. ITU –Telecommunications Standardization. VCEG-M33 (2001)
55. M. Rafie, Y. Zhang, S. Liu, Evaluation framework for video coding for machines. (2021). ISO/IEC JTC 1/SC 29/WG 2, N41
56. J. Honovich, Live Video Monitoring Usage Statistics. IPVM Report, https://ipvm.com/reports/live-video-monitoring-usage-statistics (2015-08-18)

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.