HyTIP: Hybrid Temporal Information Propagation for Masked Conditional Residual Video Coding

Yi-Hsin Chen[1] Yi-Chen Yao[1] Kuan-Wei Ho[1] Chun-Hung Wu[1] Huu-Tai Phung[1]
Martin Benjak[2] Jörn Ostermann[2] Wen-Hsiao Peng[1]

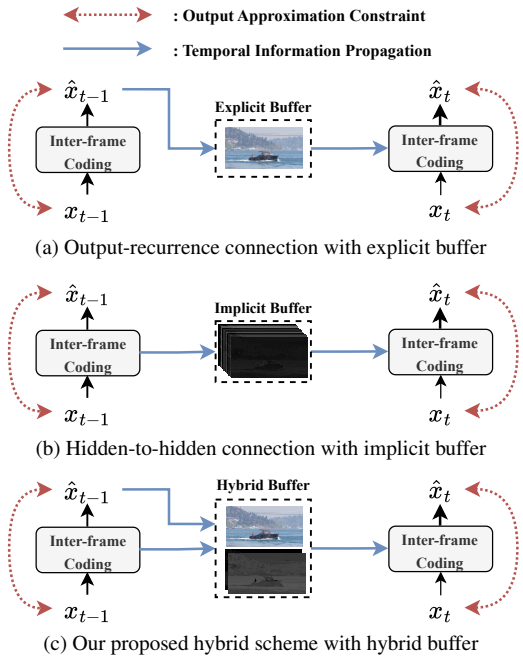
[1] National Yang Ming Chiao Tung University, Taiwan
[2] Leibniz Universität Hannover, Germany

Abstract

Most frame-based learned video codecs can be interpreted as recurrent neural networks (RNNs) propagating reference information along the temporal dimension. This work revisits the limitations of the current approaches from an RNN perspective. The output-recurrence methods, which propagate decoded frames, are intuitive but impose dual constraints on the output decoded frames, leading to suboptimal rate-distortion performance. In contrast, the hidden-to-hidden connection approaches, which propagate latent features within the RNN, offer greater flexibility but require large buffer sizes. To address these issues, we propose HyTIP, a learned video coding framework that combines both mechanisms. Our hybrid buffering strategy uses explicit decoded frames and a small number of implicit latent features to achieve competitive coding performance. Experimental results show that our HyTIP outperforms the sole use of either output-recurrence or hidden-to-hidden approaches. Furthermore, it achieves comparable performance to state-of-the-art methods but with a much smaller buffer size, and outperforms VTM 17.0 (Low-delay B) in terms of PSNR-RGB and MS-SSIM-RGB. The source code of HyTIP is available at https://github.com/NYCU-MAPL/HyTIP.

1. Introduction

Most modern learned video coding schemes bear an interpretation of implementing a recurrent neural network (RNN) along the temporal dimension. The encoding of an input video is usually performed frame-by-frame, with a shared model, composed of an encoder and a decoder, employed at each time step to convert an input video frame into its frame latents as the encoder output and to reconstruct the input frame approximately as the decoder output. The process involves leveraging the past information in the form of decoded frames and/or their latent features to encode the resulting frame latents in a rate-distortion-optimized manner. As with an RNN, the frame-by-frame coding and model weight sharing across time steps are essential for coding variable-length video sequences.



(a) Output-recurrence connection with explicit buffer

(b) Hidden-to-hidden connection with implicit buffer

(c) Our proposed hybrid scheme with hybrid buffer

Figure 1. Comparison of different temporal information propagation mechanisms in learned video compression.

Based on how temporal information is propagated through the RNN, these learned video codecs can be broadly categorized into two major approaches that feature either the output-recurrence [3, 5, 6, 8, 9, 13–19, 24, 25, 27, 28] or the hidden-to-hidden connection [21, 22, 35–37]. As depicted in Fig. 1 (a), the former approach, which adopts the output-recurrence connection, follows the same coding architecture as traditional video codecs [7, 41, 46],

where the previously decoded frame is *explicitly* buffered as a reference frame to encode the next frame. However, RNNs with only the output-recurrence connection face the challenge that the model output must closely resemble the ground truth, i.e. the input frame in the present context, while being a sufficient summary of the useful information from the past. This dual requirement leads to a compromise between preserving the past temporal information sufficiently and approximating the input frame accurately, thereby suboptimal rate-distortion performance.

To address the limitations of the output-recurrence design, the second approach aggregates temporal information by a hidden-to-hidden connection [21, 22, 31, 35–37, 47]. Instead of storing the previously decoded frame, it propagates the latent features in the RNN across time steps (see Fig. 1 (b)). In particular, the extraction and propagation of latent features are completely learned. As a result, there is little control over what information is stored, rendering the buffer's content non-explainable. To distinguish between these two buffering strategies, we refer to the buffer that stores latent features as the *implicit buffer*, in contrast to the *explicit buffer* that stores the previously decoded frame for the output-recurrence connection. Note that these temporally propagated features are not specifically constrained to approximate any input frame, although the output frame decoded from these features must closely resemble the input frame. In other words, these latent features may buffer more information from the past than is needed to reconstruct the current input frame. This design allows for more flexible temporal modeling, potentially leading to better coding efficiency. Many state-of-the-art neural video codecs belong to this category. Although some of them additionally buffer the previously decoded frame for motion estimation, the buffered frame is not directly involved as contextual information for inter-frame coding.

Methods adopting an implicit buffering strategy usually come at the cost of substantial memory requirements. For example, some [21, 22, 35–37] require the storage of at least 48 high-resolution feature maps that are of the same spatial resolution as the input frame. Generally, storing more contextual information from the past helps to improve coding efficiency. But, the data-driven nature often results in non-compact features being stored.

Recognizing the issues inherent in solely using either the output-recurrence or hidden-to-hidden connection, we propose a learned video compression framework termed HyTIP. It combines both the output-recurrence and hidden-to-hidden connections to harness the advantages of both approaches. Theoretically, RNNs that integrate both connection types are the most powerful in terms of expressivity [12]. As depicted in Fig. 1 (c), we propagate the previously decoded frame as *explicit* information along with only a small number of *implicit* latent features. Our hybrid buffering strategy leverages the prior knowledge that the previously decoded frame typically has the highest correlation with the current input frame to reduce the reliance on implicit features. By doing so, we store only a few latent features to complement the previously decoded frame. Compared to the purely implicit buffering scheme, our hybrid approach significantly reduces the buffer size while achieving similar coding performance. Moreover, compared to the explicit buffering scheme, it is able to propagate the past information that is complementary to the previously decoded frame. Our hybrid buffering strategy is validated in a masked conditional residual coding framework [9].

Table 1. Taxonomy of learned video compression works based on RNN type and the nature of buffered temporal information.

RNN Type	Buffer for Inter-frame Coding	Publications
Output-recurrence	Explicit	[3, 5, 6, 8, 9, 13–19] [24, 25, 27, 28] [34, 40, 43, 45]
Hidden-to-hidden	Implicit	[21, 22, 31, 35–37, 47]
Output-recurrence + Hidden-to-hidden	Hybrid	[11, 49]

In summary, our main contributions are as follows:

- To the best of our knowledge, this work is an early attempt to revisit the temporal information propagation mechanisms in learned video compression through an RNN perspective, analyzing their performance and buffer size trade-offs under a unified base codec.
- This work presents the first masked conditional residual coding framework that adopts a hybrid buffering strategy. This strategy propagates both implicit and explicit temporal information for motion and inter-frame coding.
- Experimental results demonstrate that our HyTIP achieves comparable coding performance to the state-of-the-art methods, but requires only 14% of their buffer size. Additionally, it outperforms VTM (Low-delay B) in terms of PSNR-RGB and MS-SSIM-RGB.
- Our simple yet efficient hybrid buffering scheme is easily extendable to other learned video codecs.

2. Related Work

2.1. Frame-based Learned Video Coding as RNN

Most learned video codecs that encode an input video frame-by-frame with an IPPP prediction structure can be thought of as a form of RNN. In RNN terminology, they propagate temporal information through either the output-recurrence or hidden-to-hidden connection. The former utilizes an *explicit* buffer to store and propagate the decoded signals from the past, such as decoded frames and/or flow maps, while the latter employs an *implicit* buffer for maintaining learned latent features. Table 1 presents a taxonomy

of the modern learned video codecs according to how they buffer and propagate temporal information.

Explicit Buffering Strategies: Under the IPPP prediction structure, many learned video codecs [3, 5, 6, 8, 9, 13–19, 24, 25, 27, 28, 34, 40, 43, 45] *explicitly* buffer the previously decoded frame(s) for inter-frame coding. Most approaches [3, 5, 6, 8, 9, 14, 15, 18, 19, 25, 27, 28, 34, 40, 43, 45] keeps only one previously decoded frame for motion estimation and inter-frame prediction because it usually correlates most strongly with the current input frame. However, some methods [24] buffer multiple decoded frames and flow maps to construct a better temporal predictor. Similar predictive coding is also found in motion coding. For example, [13, 17, 24] implement predictive motion coding based on decoded frames or flow maps. While [25, 26, 34, 40, 43, 45] propagate *implicit* features as a temporal prior for probability distribution modeling in entropy coding, these features are not directly used to generate the decoded frame. Therefore, we consider these approaches to primarily propagate contextual information in a output-recurrence manner. As discussed previously and from the perspective of the RNN structure, the dual role required of the decoded frame (or flow map) in the output-recurrence paradigm leads to sub-optimal rate-distortion performance.

Implicit Buffering Strategies: In contrast to buffering decoded frames for inter-frame coding, [35] buffers high-resolution, 64-channel latent features extracted from the inter-frame codec as contextual information for coding the next frame. In their work, when the number of feature maps stored is reduced from 64 to 9, a 2.5% BD-rate drop is observed [35]. Most follow-up works thus maintained large buffer sizes. For example, [21, 22] use 48+ channels. Along this line of research, some recent studies [36, 37] further increase the buffer size by introducing additional ConvLSTM layers [39] to propagate long-term, high-resolution temporal information, in addition to the existing 64-channel latent features. In [21, 22], the implicit buffering strategy is applied to both inter-frame and motion coding. While these methods achieve state-of-the-art coding results, their reliance on implicitly learned temporal information leads to the storage of non-compact latent features.

Hybrid Buffering Strategies: There have also been attempts [11, 49] to combine the output-recurrence and hidden-to-hidden connections with a hybrid buffering strategy. All these works *explicitly* buffer the previously decoded frame to construct a temporal predictor for inter-frame coding. Additionally, [11] introduces a ConvGRU [38] in the decoder to learn and propagate latent features for motion and inter-frame coding, while [49] use ConvLSTM layers [39] in both the encoder and the decoder by the same token. In comparison to these works, our HyTIP adopts a hybrid buffering strategy for both motion and inter-frame coding in a conditional residual coding framework. Notably, this work presents an in-depth study to shed light on the interactions between implicit and explicit buffers and their rate-distortion-complexity trade-offs.

2.2. Learned Video Coding Frameworks

There are three mainstream approaches to incorporating temporal information for inter-frame coding: (1) residual coding, (2) conditional coding, and (3) conditional residual coding. Early learned video codecs [3, 11, 15, 16, 24–28, 34, 49] predominantly follow the notion of residual coding. Similar to traditional codecs, it encodes the frame difference $x_t - x_c$ between the input frame x_t and its temporal predictor x_c generated from the propagated temporal information. Rather than employing x_c linearly, conditional coding [13, 14, 17–22, 31, 33, 35–37, 40, 42, 45, 47] encodes x_t by using x_c as a condition signal for both the encoder and decoder, allowing x_c to be utilized in a non-linear fashion for temporal prediction. This approach has been widely adopted by state-of-the-art learned video codecs. However, conditional coding could potentially suffer from the bottleneck issue [5, 6], resulting in worse coding performance than residual coding. This is most obvious when the prediction path that traverses from x_c to the decoder output is lossy and the temporal prediction is nearly perfect, i.e. $x_c \approx x_t$. To alleviate the bottleneck issue, Brand et al. [6] propose conditional residual coding, which encodes the prediction residue $x_t - x_c$ with a conditional inter-frame codec. In both lossless and lossy coding scenarios, conditional residual coding is shown to be at least as effective as conditional coding. However, this conclusion is valid under the assumption that the temporal predictor has good quality and the entropy of the residue $x_t - x_c$ is lower than that of the input frame x_t. Recognizing that these assumptions can be violated in regions with unreliable motion estimates or dis-occlusion, Chen et al. [9] propose masked conditional residual coding, where a soft mask is used to switch between conditional coding and conditional residual coding at pixel level. This work validates the hybrid buffering strategy in a masked conditional residual coding framework for its better coding efficiency. However, we stress that it is equally applicable to the other coding frameworks.

3. Hybrid Temporal Information Propagation

3.1. System Overview

Fig. 2 illustrates our HyTIP framework. It is a frame-based temporal predictive coding framework. (1) First, the motion estimation module (in green) performs motion estimation between an input frame x_t and its reference frame $\hat{x}_{t-1}$ to obtain an optical flow map f_t. (2) Second, the motion coding modules (in purple) encode f_t as $\hat{f}_t$. (3) Third, the decoded optical flow map $\hat{f}_t$ and the propagated temporal information in the buffer are used to generate temporal predic-

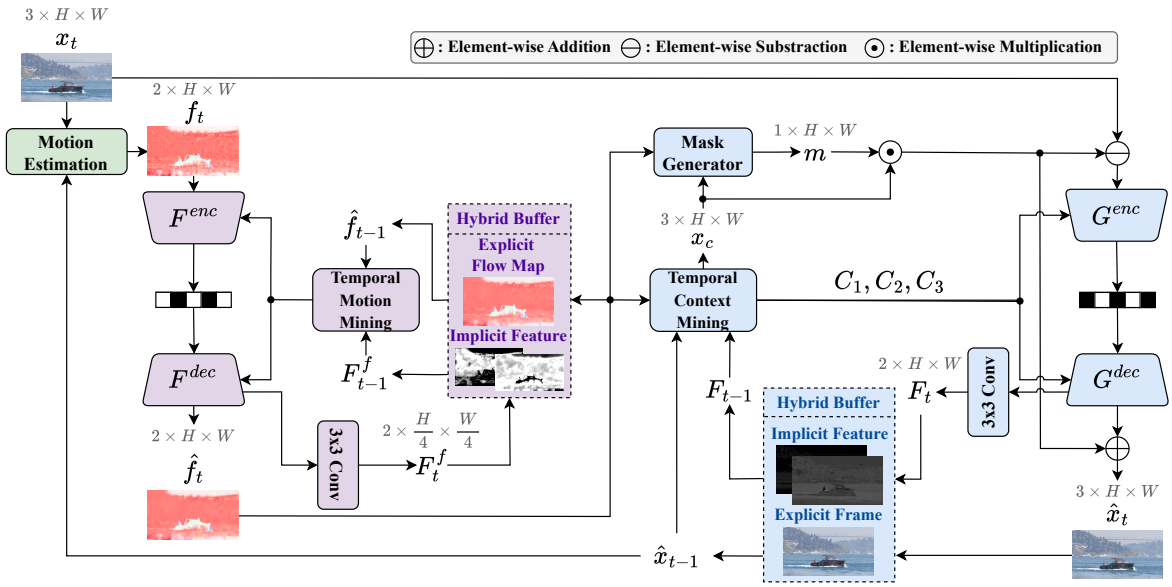


Figure 2. System overview of our proposed HyTIP. The architectures of the temporal context mining module in the inter-frame coding, the inter-frame codec $\{G^{enc}, G^{dec}\}$, and the motion codec $\{F^{enc}, F^{dec}\}$ are adapted from [22], while the mask generator follows the architecture of [9]. Further architectural details are provided in the supplementary material.

tors for x_t to assist in inter-frame coding (the blue modules). The process involves the pixel-domain predictor x_c and the multi-scale feature-domain predictors $\{C_1, C_2, C_3\}$. (4) Finally, the temporal information is buffered and propagated for coding the next input frame.

This frame-by-frame processing, along with the propagation of useful information to the next frame, can be viewed as an RNN operating along the temporal dimension. Based on how the buffered temporal information is used to generate temporal predictors, the mainstream inter-frame coding can be broadly classified into two major approaches, which implement the output-recurrence and hidden-to-hidden connections, respectively. The output-recurrence methods typically store the previous decoded frame $\hat{x}_{t-1}$ in an explicit buffer as the reference temporal information. On the other hand, the hidden-to-hidden methods buffer the latent features (i.e. F_{t-1}) from the coding process for the previous frame x_{t-1} in an implicit buffer. In this work, we integrate both the output-recurrence and hidden-to-hidden mechanisms within a unified coding framework and study the trade-offs between the coding performance and buffer size. In Fig. 2, we use a 3×3 convolutional layer to adjust the channel size of the latent features stored in the implicit buffer. Additionally we explore how various buffering strategies for motion coding impact the rate-distortion-complexity trade-offs.

Our inter-frame codec adheres to the masked conditional residual coding framework [9]. The input signal to the conditional inter-frame codec is $x_t - m \odot x_c$, where the multi-scale feature-domain temporal predictors $\{C_1, C_2, C_3\}$ serves as condition signals. Here, $\odot$ denotes element-wise multiplication, and m is a pixel-wise soft mask generated using the decoded flow map $\hat{f}_{t-1}$ and the pixel-domain temporal predictor x_c. The values of m range from 0 to 1, with all three channels of m sharing the same mask value. Since the input signal $x_t - m \odot x_c$ is equivalent to $(1 - m) \odot x_t + m \odot (x_t - x_c)$, our inter-frame codec forms a hybrid of conditional coding and conditional residual coding. Notably, we adapt the CNN-based network structure from [22], instead of using the Transformer-based coding backbone from [9]. More details are provided in the supplementary document.

3.2. Inter-frame Coding with Hybrid Buffering

Our HyTIP employs a hybrid buffering strategy that dedicates a hybrid buffer to storing both the previously decoded frame $\hat{x}_{t-1}$ and the latent features F_{t-1} for temporal prediction. As depicted in Fig. 2, these references are motion compensated with the decoded flow map $\hat{f}_t$, generating the pixel-domain temporal predictor x_c and the multi-scale features $\{C_1, C_2, C_3\}$. The predictor x_c forms a masked temporal prediction of the input frame x_t, with the prediction residues passed through the contextual encoder G^{enc} and decoder G^{dec}. Both G^{enc} and G^{dec} are conditioned on the same signals $\{C_1, C_2, C_3\}$.

In our scheme, the previously decoded frame $\hat{x}_{t-1}$ serves as the primary, but not the sole, source of temporal information for generating the temporal predictor and condition signals. The additional latent features F_{t-1} provide complementary information to enhance coding efficiency. When viewed as a form of RNN, our inter-frame codec features both the output-recurrence and hidden-to-hidden connec-

tions. It avoids the dual constraint issue, which requires $\hat{x}_{t-1}$ to be a good approximation of x_{t-1} while carrying sufficient information from the past to encode x_t. Compared to the schemes that rely solely on the hidden-to-hidden connection, where F_{t-1} is the sole source of temporal information and is learned completely from data, our hybrid buffer leverages the prior knowledge that the previously decoded frame $\hat{x}_{t-1}$ correlates highly with the current input frame x_t to reduce the reliance on the latent features F_{t-1}, thereby reducing the size of the implicit buffer. The net effect is a significant reduction in buffer size while maintaining comparable or even better coding efficiency. It is worth noting that most hidden-to-hidden-only methods inevitably buffer $\hat{x}_{t-1}$ for motion estimation, in order to propagate the latent features along the temporal dimension. However, the generation of their coded latents for x_t does not direct use $\hat{x}_{t-1}$. This design feature is in direct contrast to our scheme.

3.3. Motion Coding with Hybrid Buffering

Following the same hybrid buffering strategy as our inter-frame coding, we extend this idea to motion coding, the task of which is to encode optical flow maps. Unlike video frames, which typically contain many high-frequency details, optical flow maps are generally smoother signals along the spatial dimension, with mostly low-frequency components. Oftentimes, the optical flow evolves slowly over time in typical video sequences. Therefore, we buffer and propagate the decoded flow map $\hat{f}_{t-1}$ as an explicit temporal reference, and also the latent features F^f_{t-1} of the motion coding process for f_{t-1} as an implicit temporal reference. It is worth noting that these temporal references are used for coding f_t without motion compensation. In a manner similar to [21, 22], the latent features F^f_{t-1} are stored at one-fourth of the input resolution in both width and height. However, unlike [21, 22], which adopts a hidden-to-hidden connection approach by buffering only F^f_{t-1}, our method adopts a hybrid buffering strategy that stores both $\hat{f}_{t-1}$ and F^f_{t-1} as temporal references.

4. Experiments

4.1. Experimental Setup

Training details: We train our models with 5-frame training first on the Vimeo-90K dataset [48], consisting of 91,701 7-frame sequences, and fine-tune our models with 10-frame training on BVI-DVC [29], consisting of 800 64-frame sequences. To optimize our variable-rate model for PSNR-RGB, the hyperparameter λ, which balances rate and distortion in the objective function, is randomly sampled from [227, 2032]. Similarly, for the model optimized for MS-SSIM, λ is sampled from [7, 46]. Additional training details are provided in the supplementary material.

Evaluation Methodologies: We evaluate our method on the UVG [32], MCL-JCV [44], and HEVC Class B-E [4], and HEVC-RGB [10] datasets. Following the common test protocol for learned video coding, we convert the test sequences from YUV420 to RGB444 using BT.601. For BT.709, which is only used by DCVC-DC and DCVC-FM, we report the results in the supplementary materials. For each test sequence, the first 96 frames are encoded, and the intra period is set to 32. To ensure a fair comparison and avoid padding, the width and height of all video frames are cropped to multiples of 64. Intra coding is applied at scene cuts for all the competing methods. We report PSNR and MS-SSIM in the RGB domain and birate in bit-per-pixel (bpp). Following the common test protocol for learned video codecs, the dataset BD-rate is reported; in computing the BD-rate, a dataset-specific rate-distortion point is computed by averaging the per-frame PSNR-RGB (or MS-SSIM-RGB) and bits-per-pixel across all coded frames in the dataset. Positive and negative BD-rate numbers indicate rate inflation and reduction, respectively.

4.2. Experimental Results

This section compares our hybrid buffering strategy with the single use of the explicit or implicit strategy. For a fair assessment, we implement these buffering strategies with the same coding framework (Fig. 2). Moreover, we align the buffer size for the hybrid and implicit strategies. As an example, if the hybrid variant keeps one previously decoded frame (3 channels) and two latent feature maps, then the implicit variant maintains five latent feature maps. To reduce the training time required for extensive experiments, we disable the hierarchical quality structure [22], the channel transform module [9], the checkerboard context model [23] in the inter-frame codec, and the variable-rate modules, as opposed to the full model used in Section 4.3. More architectural details are provided in the supplementary material.

Explicit, Implicit, and Hybrid Buffering: Table 2 compares the coding performance of adopting various buffering strategies for motion and inter-frame coding. For this experiment, we train the competing methods on 5-frame training sequences in Vimeo-90K only. From Table 2, we make the following observations:

(1) *Robustness of propagating latent features with a large buffer*: As expected, both the hybrid and implicit variants with a large buffer size outperform the explicit one in both motion and inter-frame coding, as they can propagate a larger number of latent feature maps, using more effectively temporal information.

(2) *Sensitivity of implicit buffering to buffer size reduction:* The coding performance of the implicit buffering strategy, when applied to motion or inter-frame coding, decreases significantly with reduced buffer size. With the implicit strategy, the model solely relies on learning, in order

Table 2. BD-rate (%) comparison of different buffering strategies with different buffer sizes. The anchor employs explicit buffering in both motion and inter-frame coding. The values in parentheses indicate the number of full-resolution feature maps buffered for coding one input flow map or frame (explicit + implicit). One flow map and one RGB frame are equivalent to two and three full-resolution feature maps, respectively. For clarity, buffer sizes for entropy coding are excluded here, in contrast to Section 4.3.

Motion	Inter	UVG	MCL-JCV	HEVC-B	HEVC-C	HEVC-D	HEVC-E	HEVC-RGB	Average
Explicit (2+0)	Explicit (3+0)	0	0	0	0	0	0	0	0
Implicit (0+4)	Explicit (3+0)	-11.5	-9.5	-12.6	-18.6	-12.6	-11.9	-8.8	-12.2
Implicit (0+2.125)	Explicit (3+0)	-6.5	-3.8	-6.2	-10.4	-6.7	-8.6	-3.0	-6.5
Hybrid (2+4)	Explicit (3+0)	-15.1	-13.2	-15.4	-23.8	-19.9	-15.8	-9.1	-16.0
Hybrid (2+0.125)	Explicit (3+0)	-13.0	-11.2	-14.1	-19.1	-16.3	-18.1	-11.0	-14.7
Hybrid (2+0.125)	Implicit (0+51)	-12.6	-15.5	-20.1	-30.2	-27.8	-20.6	-10.5	-19.6
Hybrid (2+0.125)	Implicit (0+5)	-9.2	-13.0	-14.9	-24.6	-21.4	-14.6	-7.4	-15.0
Hybrid (2+0.125)	Hybrid (3+48)	-17.3	-16.3	-21.0	-30.3	-27.3	-25.7	-15.1	-21.9
Hybrid (2+0.125)	Hybrid (3+2)	-17.6	-15.9	-20.3	-29.1	-25.9	-26.6	-14.8	-21.5

Table 3. BD-rate (%) comparison of longer sequence training impact on three buffering strategies for inter-frame coding. The anchor is the variant employing explicit buffering in both motion and inter-frame coding. The values in parentheses, as in Table 2, indicate the number of full-resolution feature maps buffered for coding one input flow map or frame (explicit + implicit).

Motion	Inter	# Frame	UVG	MCL-JCV	HEVC-B	HEVC-C	HEVC-D	HEVC-E	HEVC-RGB	Average
Hybrid (2+0.125)	Explicit (3+0)	5	-13.0	-11.2	-14.1	-19.1	-16.3	-18.1	-11.0	-14.7
		10	-19.9	-16.3	-16.1	-20.9	-18.4	-20.2	-13.2	-17.9
Hybrid (2+0.125)	Implicit (0+5)	5	-9.2	-13.0	-14.9	-24.6	-21.4	-14.6	-7.4	-15.0
		10	-21.4	-22.0	-21.0	-30.2	-25.2	-13.2	-12.6	-20.8
Hybrid (2+0.125)	Hybrid (2+3)	5	-17.6	-15.9	-20.3	-29.1	-25.9	-26.7	-14.8	-21.5
		10	-25.4	-21.7	-25.1	-34.5	-30.6	-29.1	-20.3	-26.7

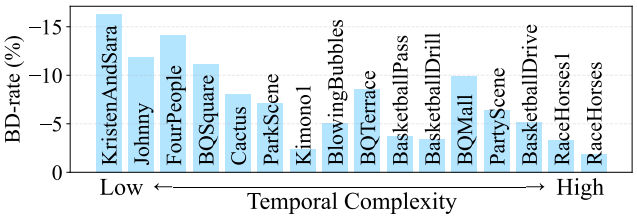


Figure 3. Per-sequence BD-rate results of the inter-frame codec using hybrid versus implicit buffering (anchor) on the HEVC Class B-E datasets, both under the small buffer setting in Table 2. Results are ordered by sequence temporal complexity [30].

to extract useful temporal information. It is challenging to train a model that is capable of extracting both useful and compact temporal features.

(3) *Resilience of hybrid buffering to buffer size reduction:* In contrast, our hybrid buffering strategy with small buffer size performs comparable to the large buffer size variant, with a reduced buffer size for latent features in either motion or inter-frame codec. Unlike the implicit strategy, our hybrid strategy relies heavily on the previously decoded frame as the primary source for temporal prediction. The latent features play a secondary role in complementing the previously decoded frame, which generally has the highest correlation with the current input frame.

(4) *Superiority of hybrid buffering with a small buffer:* Compared to the explicit strategy, our hybrid strategy with a small buffer size achieves notable gains in both motion and inter-frame coding (14.7% and additional 6.8% BD-rate savings, respectively). It is to be noted that the hybrid buffering uses a slightly larger buffer size for motion and inter-frame coding (2 additional feature maps of one-fourth the input spatial resolution, which is equivalent to 0.125 channel of the input resolution feature, for motion coding, and likewise, 2 additional feature maps of the input resolution for inter-frame coding). The explicit strategy underperforms due mainly to the dual constraint issue (see Fig. 1), where the decoded frame must meet two requirements: (1) closely approximating the input frame and (2) serving as a

Table 4. Complexity comparison of encoding/decoding MACs and model size across buffering strategies. The values in parentheses, as in Table 2, indicate the number of full-resolution feature maps buffered for coding one input flow map or frame (explicit + implicit).

Motion	Inter	Model Size (M)	Enc. / Dec. kMACs/pixel
Explicit (2+0)	Explicit (3+0)	13.712	1277 / 858
Implicit (0+4)	Explicit (3+0)	13.358	1268 / 849
Implicit (2+0.125)	Explicit (3+0)	13.323	1266 / 847
Hybrid (2+4)	Explicit (3+0)	13.877	1284 / 865
Hybrid (2+0.125)	Explicit (3+0)	13.805	1280 / 861
Hybrid (2+0.125)	Implicit (0+51)	14.918	1303 / 884
Hybrid (2+0.125)	Implicit (0+5)	14.891	1281 / 862
Hybrid (2+0.125)	Hybrid (3+48)	14.917	1302 / 883
Hybrid (2+0.125)	Hybrid (2+3)	14.890	1280 / 861

Table 5. BD-rate (%) comparison between HyTIP and the state-of-the-art methods in terms of PSNR-RGB. The anchor is VTM 17.0.

	UVG	MCL-JCV	HEVC-B	HEVC-C	HEVC-D	HEVC-E	HEVC-RGB	Average
VTM 17.0 [2]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
HM 16.25 [1]	26.3	36.8	31.8	29.8	29.6	32.5	34.0	31.5
MaskCRT [9]	-10.1	7.4	2.7	21.6	-6.4	17.1	-3.4	4.1
DCVC-TCM [35]	16.1	27.4	29.3	59.8	18.8	61.3	24.3	33.9
DCVC-HEM [20]	-19.2	-8.5	-4.4	14.9	-15.0	2.5	-11.0	-5.8
DCVC-DC [21]	-29.9	-21.4	-16.5	-9.4	-30.3	-28.1	-29.7	-23.6
DCVC-FM [22]	-23.9	-13.4	-10.9	-5.4	-26.9	-29.2	-19.7	-18.5
HyTIP (Ours)	-32.2	-15.7	-19.6	-12.0	-32.4	-18.6	-24.5	-22.1

Table 6. BD-rate (%) comparison between HyTIP and the state-of-the-art methods in terms of MS-SSIM-RGB. The anchor is VTM 17.0.

	UVG	MCL-JCV	HEVC-B	HEVC-C	HEVC-D	HEVC-E	HEVC-RGB	Average
VTM [2]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
HM [1]	20.2	30.2	27.4	27.8	29.0	28.8	27.2	27.2
MaskCRT [9]	-27.0	-36.5	-44.9	-34.1	-48.5	-39.3	-42.9	-39.0
DCVC-TCM [35]	-12.1	-25.3	-31.3	-21.6	-38.8	-29.9	-25.9	-26.4
DCVC-HEM [20]	-31.5	-45.0	-50.6	-43.3	-57.5	-55.7	-45.6	-47.0
DCVC-DC [21]	-37.3	-50.9	-56.8	-54.4	-64.9	-66.0	-55.8	-55.2
HyTIP (Ours)	-42.8	-53.4	-59.7	-54.8	-64.4	-64.0	-57.8	-56.7

sufficient summary of past information for propagation. In comparison, the implicit strategy with a small buffer size shows some improvement over the explicit one (6.5% and additional 0.3% BD-rate savings, respectively). However, its gain is not as significant as that of our hybrid approach due to its purely learning-based nature, which fails to take advantage of the prior knowledge that the previously decoded frame correlates highly with the current input frame.

Fig. 3 further presents the per-sequence BD-rate savings of hybrid buffering strategy to the implicit buffering strategy. As shown, the hybrid strategy has greater gains than the implicit strategy in sequences with lower temporal complexity. This is likely because the hybrid strategy allows the model to directly use the previous decoded frame, which typically has the highest correlation with the current frame.

Training with longer sequences: Generally, training with longer sequences can enhance the learning of RNNs by allowing them to capture long-term dependencies. Table 3 analyzes the impact of long-sequence training on the three buffering strategies in the context of inter-frame coding. Specifically, we fine-tune each initial codec, originally trained on Vimeo-90K [48] with 5-frame sequences, using BVI-DVC [29] with 10-frame sequences. From Table 3, all three approaches benefit from long-sequence training. However, the explicit buffering strategy achieves a performance gain of only 3.2%, which is lower than the 5.8% and 5.2% gains observed in the implicit and hybrid buffering strategies, respectively. This discrepancy arises from the dual constraint on the output frame in explicit buffering. In contrast, the implicit and hybrid buffering strategies, which propagate implicit features without such constraints, are better able to leverage long-sequence training.

Complexity comparison: Table 4 presents the complexity comparison of various buffering strategies in terms of the model size, kilo-multiply-accumulate operations per pixel (kMACs/pixel), and buffer size. As reported in Table 2, our hybrid scheme has the best coding performance under a similar buffer size, while Table 4 further confirms that its model size and MAC are comparable to the other schemes. The implicit and hybrid buffering strategies introduce only two minor structural modifications compared to the explicit buffering strategy: (1) an additional CNN to adjust the channel size of the buffered implicit features in the motion and inter-frame codecs and (2) a slight change to the first-layer convolution in temporal context mining (or temporal motion mining) for generating inter-frame (or motion) temporal predictors (see Fig. 2). As a result, the increase in the model size and encoding/decoding kMAC/pixel is negligible. Although our hybrid buffering strategy with a small buffer size requires storing additionally 0.125 and 2 channels of full-resolution implicit features for motion and inter-frame coding, respectively, compared to the explicit buffering strategy, it still requires significantly fewer channels than the 48+ implicit features used in state-of-the-art methods [20–22, 35]. To put things into context, traditional codecs typically buffer four reference frames.

4.3. Comparison with the State-of-the-Arts

This section compares our method with state-of-the-art traditional codecs and learned P-frame codecs, in terms of rate-distortion performance. The traditional codecs include HM 16.25 [1] and VTM 17.0 [2] under the low-delay-B configuration. Following [21], both HM and VTM are configured to encode the input video in YUV444 as the internal color space. This setting is shown to achieve better coding results than using YUV420 when the final distortion is measured in the RGB domain. The learned P-frame codecs include DCVC-TCM [35], DCVC-HEM [20], DCVC-DC [21], and DCVC-FM [22], which adopt the im-

Table 7. Complexity comparison between HyTIP and the state-of-the-art learned P-frame codecs. Buffer size represents the number of full-resolution feature maps that need to be buffered for coding one input frame.

Methods	Model Size (M)	Enc. / Dec. kMACs/pixel	Buffer Size
DCVC-TCM [35]	10.709	1406 / 917	67
DCVC-HEM [20]	17.523	1662 / 1243	67.625
DCVC-DC [21]	19.779	1343 / 918	55.75
DCVC-FM [22]	18.336	1137 / 871	55.75
MaskCRT [9]	31.152	1401 / 763	13
HyTIP (Ours)	16.593	1293 / 873	7.875

plicit buffering strategy, as well as MaskCRT [9], which employs the explicit buffering strategy.

Tables 5 and 6 present the coding performance of the competing methods in terms of PSNR and MS-SSIM, respectively. As shown, our HyTIP achieves comparable coding performance to the state-of-the-art method DCVC-DC [21], except on HEVC-E, which is a special dataset that consists of video conferencing-type sequences with static backgrounds. Aside from DCVC-DC, HyTIP outperforms the other methods, including traditional codecs HM 16.25 [1] and VTM 17.0 [2].

It is important to note that the training procedure significantly impacts the coding performance of a neural video codec. Since the state-of-the-art DCVC-DC [21] and DCVC-FM [22] have not released their training details, we are unable to reproduce their results for a fair comparison. The results shown here are obtained with their publicly available test models. However, this work adopts the training procedure from [9], which is different from those for DCVC-DC [21] and DCVC-FM [22]. Consequently, even though our HyTIP achieves performance similar to DCVC-DC [21] or slightly worse on HEVC-E, this does not imply that our hybrid buffering strategy is ineffective. According to the experiments in Section 4.2, there is still potential to combine their approaches with ours to achieve further gains.

Table 7 compares the complexity of the competing methods in terms of the model size, kMACs/pixel, and buffer size. Fig. 4 reports how these methods trade off between complexity and BD-rate. As shown, HyTIP requires the smallest buffer size among all the competing methods, utilizing only 14% of the buffer size required by works using the implicit buffering strategy, such as DCVC-TCM [35], DCVC-HEM [20], DCVC-DC [21], and DCVC-FM [22]. Additionally, the decoding kMAC/pixel of HyTIP is comparable to DCVC-FM [22] and slightly lower than DCVC-DC [21]. Note that buffer size reflects the amount of data that must be fetched at each time step for encoding/decoding a video frame. Typically, the buffer stores previously decoded frames and/or features for temporal prediction resides in off-chip memory. A larger buffer size in this context requires fetching more data from off-chip memory,

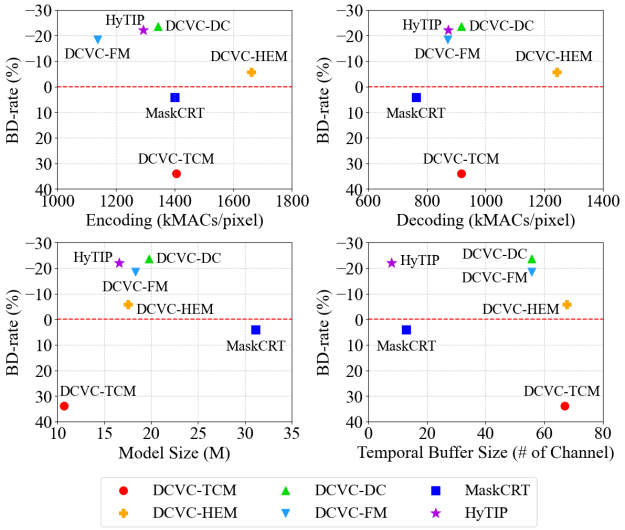


Figure 4. Comparison of complexity-performance trade-offs between HyTIP and the state-of-the art methods. The vertical axis is the BD-rate savings in terms of PSNR-RGB evaluated with VTM-17.0 (low delay B) serving as the anchor. Positive and negative BD-rate numbers indicate rate inflation and reduction, respectively. The horizontal axis is the complexity metrics.

leading to higher memory bandwidth demands and reduced practicality. A similar design constraint was adopted in traditional video codecs, where block prediction typically relies on only two reference blocks, motivates our focus on reducing buffer size. While large buffer size is intrinsic to the algorithm design, both the model size and kMAC can be further reduced through network optimization.

5. Conclusion

In this work, we revisit how typical learned video compression frameworks propagate temporal information to assist coding from an RNN perspective, and propose HyTIP, combining output-recurrence and hidden-to-hidden connections to leverage the advantages of both approaches. By propagating the previously decoded frame as explicit information to primarily serve as temporal data, we only require a small number of implicit latent features to carry complementary temporal information, achieving competitive performance. Our experimental results confirm the superiority of the combined use of implicit and explicit buffering strategies over the use of either alone. This hybrid approach is less sensitive to buffer size than purely implicit buffering strategies. Compared to state-of-the-art methods that adopt solely implicit buffering strategy, our HyTIP requires only 14% of the buffer size while achieving comparable performance on most testsets. In this work, we aim to investigate the design of temporal propagation mechanisms within learned video codecs, rather than their functionality. Extending our codec to enhance its functionality, such as YUV coding, is among our future directions.

Acknowledgement

This work is supported by MediaTek and National Science and Technology Council (NSTC), Taiwan, under Grants 113-2634-F-A49-007- and 111-2923-E-A49-007-MY3. We thank National Center for High-performance Computing (NCHC) for providing computational and storage resources, and NVAITC for providing access to the Taipei-1 supercomputer.

References

[1] https://vcgit.hhi.fraunhofer.de/jvet/HM/. 7, 8, 14

[2] https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM. 7, 8, 12, 14

[3] Eirikur Agustsson, David Minnen, Nick Johnston, Johannes Ballé, Sung Jin Hwang, and George Toderici. Scale-space flow for end-to-end optimized video compression. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8500–8509, 2020. 1, 2, 3

[4] Frank Bossen et al. Common test conditions and software reference configurations. *JCTVC-L1100*, 12(7), 2013. 5

[5] Fabian Brand, Jürgen Seiler, and André Kaup. On benefits and challenges of conditional interframe video coding in light of information theory. In *2022 Picture Coding Symposium (PCS)*, pages 289–293. IEEE, 2022. 1, 2, 3

[6] Fabian Brand, Jürgen Seiler, and André Kaup. Conditional residual coding: A remedy for bottleneck problems in conditional inter frame coding. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 34(7):6445–6459, 2024. 1, 2, 3

[7] Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J. Sullivan, and Jens-Rainer Ohm. Overview of the versatile video coding (vvc) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 31(10):3736–3764, 2021. 1

[8] Yi-Hsin Chen, Kuan-Wei Ho, Martin Benjak, Jörn Ostermann, and Wen-Hsiao Peng. On the rate-distortion-complexity trade-offs of neural video coding. In *2024 IEEE 26th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6, 2024. 1, 2, 3

[9] Yi-Hsin Chen, Hong-Sheng Xie, Cheng-Wei Chen, Zong-Lin Gao, Martin Benjak, Wen-Hsiao Peng, and Jörn Ostermann. Maskcrt: Masked conditional residual transformer for learned video compression. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 34(11):11980–11992, 2024. 1, 2, 3, 4, 5, 7, 8, 11, 14, 19

[10] D Flynn et al. Common test conditions and software reference configurations for hevc range extensions. *JCVTC-n1006*, 2013. 5

[11] Adam Golinski, Reza Pourreza, Yang Yang, Guillaume Sautiere, and Taco S Cohen. Feedback recurrent autoencoder for video compression. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 2020. 2, 3

[12] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. MIT press Cambridge, 2016. 2

[13] Yung-Han Ho, Chih-Peng Chang, Peng-Yu Chen, Alessandro Gnutti, and Wen-Hsiao Peng. Canf-vc: Conditional augmented normalizing flows for video compression. In *European Conference on Computer Vision (ECCV)*, pages 207–223, 2022. 1, 2, 3

[14] Yung-Han Ho, Chih-Hsuan Lin, Peng-Yu Chen, Mu-Jung Chen, Chih-Peng Chang, Wen-Hsiao Peng, and Hsueh-Ming Hang. Learned video compression for yuv 4: 2: 0 content using flow-based conditional inter-frame coding. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 829–833. IEEE, 2022. 3

[15] Zhihao Hu, Zhenghao Chen, Dong Xu, Guo Lu, Wanli Ouyang, and Shuhang Gu. Improving deep video compression by resolution-adaptive flow coding. In *European Conference on Computer Vision (ECCV)*, pages 193–209, 2020. 3

[16] Zhihao Hu, Guo Lu, and Dong Xu. Fvc: A new framework towards deep video compression in feature space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1502–1511, 2021. 3

[17] Zhihao Hu, Guo Lu, Jinyang Guo, Shan Liu, Wei Jiang, and Dong Xu. Coarse-to-fine deep video coding with hyperprior-guided mode prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5921–5930, 2022. 3

[18] Théo Ladune, Pierrick Philippe, Wassim Hamidouche, Lu Zhang, and Olivier Déforges. Modenet: Mode selection network for learned video coding. In *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2020. 3

[19] Jiahao Li, Bin Li, and Yan Lu. Deep contextual video compression. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 18114–18125, 2021. 1, 2, 3

[20] Jiahao Li, Bin Li, and Yan Lu. Hybrid spatial-temporal entropy modelling for neural video compression. In *Proceedings of the 30th ACM International Conference on Multimedia (ACM MM)*, pages 1503–1511, 2022. 7, 8, 14

[21] Jiahao Li, Bin Li, and Yan Lu. Neural video compression with diverse contexts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22616–22626, 2023. 1, 2, 3, 5, 7, 8, 11, 12, 14

[22] Jiahao Li, Bin Li, and Yan Lu. Neural video compression with feature modulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26099–26108, 2024. 1, 2, 3, 4, 5, 7, 8, 11, 14, 19

[23] Fangzheng Lin, Heming Sun, Jinming Liu, and Jiro Katto. Multistage spatial context models for learned image compression. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023. 5, 11

[24] Jianping Lin, Dong Liu, Houqiang Li, and Feng Wu. M-lvc: Multiple frames prediction for learned video compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 3546–3554, 2020. 1, 2, 3

[25] Haojie Liu, Ming Lu, Zhan Ma, Fan Wang, Zhihuang Xie, Xun Cao, and Yao Wang. Neural video coding using multiscale motion compensation and spatiotemporal context model. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 31(8):3182–3196, 2020. 1, 2, 3

[26] Haojie Liu, Han Shen, Lichao Huang, Ming Lu, Tong Chen, and Zhan Ma. Learned video compression via joint spatial-temporal correlation exploration. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 11580–11587, 2020. 3

[27] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11006–11015, 2019. 1, 2, 3

[28] Guo Lu, Chunlei Cai, Xiaoyun Zhang, Li Chen, Wanli Ouyang, Dong Xu, and Zhiyong Gao. Content adaptive and error propagation aware deep video compression. In *European Conference on Computer Vision (ECCV)*, pages 456–472, 2020. 1, 2, 3, 19

[29] Di Ma, Fan Zhang, and David R Bull. Bvi-dvc: A training database for deep video compression. *IEEE Transactions on Multimedia*, 24:3847–3858, 2021. 5, 7

[30] Vignesh V Menon, Christian Feldmann, Hadi Amirpour, Mohammad Ghanbari, and Christian Timmerer. Vca: video complexity analyzer. In *Proceedings of the 13th ACM multimedia systems conference*, pages 259–264, 2022. 6

[31] Fabian Mentzer, George Toderici, David Minnen, Sergi Caelles, Sung Jin Hwang, Mario Lucic, and Eirikur Agustsson. VCT: A video compression transformer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2, 3

[32] Alexandre Mercat, Marko Viitanen, and Jarno Vanne. UVG dataset: 50/120fps 4k sequences for video codec analysis and development. In *Proceedings of the ACM Multimedia Systems Conference*, pages 297–302, 2020. 5

[33] Linfeng Qi, Jiahao Li, Bin Li, Houqiang Li, and Yan Lu. Motion information propagation for neural video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6111–6120, 2023. 3

[34] Oren Rippel, Sanjay Nair, Carissa Lew, Steve Branson, Alexander G Anderson, and Lubomir Bourdev. Learned video compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3454–3463, 2019. 2, 3

[35] Xihua Sheng, Jiahao Li, Bin Li, Li Li, Dong Liu, and Yan Lu. Temporal context mining for learned video compression. *IEEE Transactions on Multimedia (TMM)*, 25:7311–7322, 2023. 1, 2, 3, 7, 8, 14

[36] Xihua Sheng, Li Li, Dong Liu, and Houqiang Li. Prediction and reference quality adaptation for learned video compression. *arXiv preprint arXiv:2406.14118*, 2024. 3

[37] Xihua Sheng, Li Li, Dong Liu, and Houqiang Li. Spatial decomposition and temporal fusion based inter prediction for learned video compression. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 34(7):6460–6473, 2024. 1, 2, 3

[38] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems (NeurIPS)*, 28, 2015. 3

[39] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems (NeurIPS)*, 28, 2015. 3

[40] Yibo Shi, Yunying Ge, Jing Wang, and Jue Mao. Alphavc: High-performance and efficient learned video compression. In *European Conference on Computer Vision (ECCV)*, pages 616–631, 2022. 2, 3

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

[42] Feng Wang, Haihang Ruan, Fei Xiong, Jiayu Yang, Litian Li, and Ronggang Wang. Butterfly: Multiple reference frames feature propagation mechanism for neural video compression. In *2023 Data Compression Conference (DCC)*, pages 198–207. IEEE, 2023. 3

[43] Huairui Wang and Zhenzhong Chen. Exploring long- and short-range temporal information for learned video compression. *IEEE Transactions on Image Processing (TIP)*, 33:780–792, 2024. 2, 3

[44] Haiqiang Wang, Weihao Gan, Sudeng Hu, Joe Yuchieh Lin, Lina Jin, Longguang Song, Ping Wang, Ioannis Katsavounidis, Anne Aaron, and C-C Jay Kuo. MCL-JCV: a jnd-based h. 264/avc video quality assessment dataset. In *IEEE International Conference on Image Processing (ICIP)*, pages 1509–1513, 2016. 5

[45] Huairui Wang, Nianxiang Fu, and Zhenzhong Chen. Efficient learned video compression via bidirectional temporal information exploration. In *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2023. 2, 3

[46] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 13(7):560–576, 2003. 1

[47] Jinxi Xiang, Kuan Tian, and Jun Zhang. Mimt: Masked image modeling transformer for video compression. In *International Conference on Learning Representations (ICLR)*, 2022. 2, 3

[48] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision (IJCV)*, 127(8):1106–1125, 2019. 5, 7

[49] Ren Yang, Fabian Mentzer, Luc Van Gool, and Radu Timofte. Learning for video compression with recurrent auto-encoder and recurrent probability model. *IEEE Journal of Selected Topics in Signal Processing (JSTSP)*, 15(2):388–401, 2020. 2, 3

HyTIP: Hybrid Temporal Information Propagation for Masked Conditional Residual Video Coding
Supplementary Materials

Yi-Hsin Chen[1] Yi-Chen Yao[1] Kuan-Wei Ho[1] Chun-Hung Wu[1] Huu-Tai Phung[1]
Martin Benjak[2] Jörn Ostermann[2] Wen-Hsiao Peng[1]

[1] National Yang Ming Chiao Tung University, Taiwan
[2] Leibniz Universität Hannover, Germany

This supplementary document provides the following additional materials and results to assist with the understanding of our HyTIP.

- Additional results on buffer size in Section A1;
- Results on long-sequence training under BT.709 color space conversion in Section A2;
- Rate-distortion comparisons to state-of-the-arts in Section A3;
- Network architecture details in Section A4;
- Training details in Section A5;
- Configurations of HM 16.25 and VTM 17.0 in Section A6;

A1. Additional Results on Buffer Size

Table A1 and Table A2 provide additional results on the buffer size configurations reported in Table 2 of the main paper, using BT.601 and BT.709 color space conversions, respectively. As shown, the conclusions are consistent with those in the main paper, where the hybrid buffering strategy demonstrates strong robustness and efficiency. It achieves better performance than both the explicit and implicit buffering strategies and exhibits greater resilience to buffer size reduction compared to the implicit buffering strategy. According to the results in Table A1 and Table A2, our final design buffers 1 explicit decoded flow map and 0.125 full-resolution feature map (i.e., 2+0.125 full-resolution feature maps) for motion coding, and 1 explicit decoded frame and 2 full-resolution feature maps (i.e., 3+2 full-resolution feature maps) for inter-frame coding to balance coding performance and complexity.

A2. Results on Long-sequence Training under BT.709 Color Space Conversion

Table A3 presents the results of the same study as Table 3 in the main paper, but employs BT.709 color space conversion, following [21, 22], instead of BT.601 for the YUV420 to RGB444 conversion. As shown, the conclusions remain consistent with those in the main paper, where training with longer sequences generally improves all buffering strategies. The explicit buffering strategy shows a smaller performance gain due to its dual constraint on the output frame, while the implicit and hybrid strategies, which propagate implicit features without such constraints, are better able to leverage long-sequence training.

A3. Rate-distortion Comparisons to State-of-the-arts

Fig. A1 and Fig. A2 present the rate-distortion comparisons between our method and state-of-the-art approaches in terms of PSNR-RGB, using BT.601 and BT.709 for color space conversion, respectively. Similarly, Fig. A3 and Fig. A4 show the comparisons in terms of MS-SSIM-RGB under BT.601 and BT.709 color space conversion, respectively. The corresponding BD-rates using BT.709 as the color space conversion are summarized in Table A4 and Table A5.

A4. Network Architecture Details

Fig. A5, Fig. A6, and Fig. A7 provide the network architecture details in Fig. 2 of the main paper. The base motion and inter-frame codec are adapted from [22] while the channel transform module in the inter-frame codec is from [9] and the checkerboard context model in the inter-frame codec is from [23].

A5. Training Details

Table A6 summarizes our HyTIP training procedure, adapted from [9]. The first six phases follow [9], using explicit temporal reference $\hat{x}_{t-1}$ as the temporal reference in-

formation only in the inter-frame codec. Subsequently, the implicit related module is incorporated for further training.

A6. Configurations of HM 16.25 and VTM 17.0

Following the recommendation from [21], we encode videos in YUV444 format. We use the *encoder_lowdelay_vtm.cfg* of VTM [2] with the following parameters:

–c {config file name}
–InputFile={input file name}
–InputBitDepth=8
–InputChromaFormat=444
–ChromaFormatIDC=444
–InternalBitDepth=10
–OutputBitDepth=8
–DecodingRefreshType=2
–FrameRate={frame rate}
–FrameSkip=0
–SourceWidth={width}
–SourceHeight={height}
–FramesToBeEncoded=96
–Level=4.1
–IntraPeriod=32
–QP={qp}
–BitstreamFile={bitstream file name}
–ReconFile={reconstruction file name}

Similarly, We use the *encoder_lowdelay_main_rext.cfg* of HM [2] with the following parameters:

–c {config file name}
–InputFile={input file name}
–InputBitDepth=8
–InputChromaFormat=444
–ChromaFormatIDC=444
–InternalBitDepth=10
–InternalBitDepthC=10
–OutputBitDepth=8
–OutputBitDepthC=8
–FrameRate={frame rate}
–FrameSkip=0
–SourceWidth={width}
–SourceHeight={height}
–FramesToBeEncoded=96
–Level=4.1
–IntraPeriod=32
–QP={qp}
–BitstreamFile={bitstream file name}
–ReconFile={reconstruction file name}

Table A1. BD-rate (%) comparison of different buffering strategies with different buffer sizes, using BT.601 for color space conversion. The anchor employs explicit buffering in both motion and inter-frame coding. The values in parentheses, as in Table 2, indicate the number of full-resolution feature maps buffered for coding one input flow map or frame (explicit + implicit).

Motion	Inter	UVG	MCL-JCV	HEVC-B	HEVC-C	HEVC-D	HEVC-E	HEVC-RGB	Average
Explicit (2+0)	Explicit (3+0)	0	0	0	0	0	0	0	0
Implicit (0+4)	Explicit (3+0)	-11.5	-9.5	-12.6	-18.6	-12.6	-11.9	-8.8	-12.2
Implicit (0+2.1875)	Explicit (3+0)	-7.0	-3.6	-8.7	-12.0	-8.4	-9.8	-7.2	-8.1
Implicit (0+2.125)	Explicit (3+0)	-6.5	-3.8	-6.2	-10.4	-6.7	-8.6	-3.0	-6.5
Implicit (0+2.0625)	Explicit (3+0)	-4.0	2.8	-1.9	-2.2	0.8	-2.3	-2.3	-1.3
Hybrid (2+4)	Explicit (3+0)	-15.1	-13.2	-15.4	-23.8	-19.9	-15.8	-9.1	-16.0
Hybrid (2+0.1875)	Explicit (3+0)	-13.6	-9.6	-14.8	-21.4	-18.0	-18.4	-13.1	-15.6
Hybrid (2+0.125)	Explicit (3+0)	-13.0	-11.2	-14.1	-19.1	-16.3	-18.1	-11.0	-14.7
Hybrid (2+0.0625)	Explicit (3+0)	-9.1	-5.4	-9.3	-13.9	-11.7	-13.4	-6.7	-9.9
Hybrid (2+0.125)	Implicit (0+51)	-12.6	-15.5	-20.1	-30.2	-27.8	-20.6	-10.5	-19.6
Hybrid (2+0.125)	Implicit (0+6)	-10.0	-12.6	-15.8	-24.5	-21.9	-13.0	-8.6	-15.2
Hybrid (2+0.125)	Implicit (0+5)	-9.2	-13.0	-14.9	-24.6	-21.4	-14.6	-7.4	-15.0
Hybrid (2+0.125)	Implicit (0+4)	-5.9	-9.0	-13.5	-19.5	-17.6	-1.9	-4.3	-10.2
Hybrid (2+0.125)	Hybrid (3+48)	-17.3	-16.3	-21.0	-30.3	-27.3	-25.7	-15.1	-21.9
Hybrid (2+0.125)	Hybrid (3+3)	-16.5	-15.7	-20.8	-29.1	-26.8	-23.3	-14.7	-21.0
Hybrid (2+0.125)	Hybrid (3+2)	-17.6	-15.9	-20.3	-29.1	-25.9	-26.6	-14.8	-21.5
Hybrid (2+0.125)	Hybrid (3+1)	-12.7	-10.4	-14.8	-23.6	-19.3	-18.9	-9.1	-15.5

Table A2. BD-rate (%) comparison of different buffering strategies with different buffer sizes, using BT.709 for color space conversion. The anchor employs explicit buffering in both motion and inter-frame coding. The values in parentheses, as in Table 2, indicate the number of full-resolution feature maps buffered for coding one input flow map or frame (explicit + implicit).

Motion	Inter	UVG	MCL-JCV	HEVC-B	HEVC-C	HEVC-D	HEVC-E	HEVC-RGB	Average
Explicit (2+0)	Explicit (3+0)	0	0	0	0	0	0	0	0
Implicit (0+4)	Explicit (3+0)	-10.0	-8.1	-10.5	-14.5	-9.0	-10.8	-8.8	-10.2
Implicit (0+2.1875)	Explicit (3+0)	-6.5	-5.9	-8.0	-10.6	-6.0	-9.4	-7.2	-7.7
Implicit (0+2.125)	Explicit (3+0)	-5.5	-3.6	-5.8	-9.3	-5.1	-7.2	-3.0	-5.6
Implicit (0+2.0625)	Explicit (3+0)	-3.7	-0.1	-1.6	-2.2	1.1	-1.5	-2.3	-1.5
Hybrid (2+4)	Explicit (3+0)	-13.3	-11.5	-12.8	-19.1	-15.0	-14.4	-9.1	-13.6
Hybrid (2+0.1875)	Explicit (3+0)	-12.9	-11.4	-13.6	-18.3	-14.6	-17.0	-13.1	-14.4
Hybrid (2+0.125)	Explicit (3+0)	-12.2	-10.3	-12.4	-15.5	-12.2	-18.7	-11.0	-13.2
Hybrid (2+0.0625)	Explicit (3+0)	-8.7	-6.7	-8.3	-10.7	-7.8	-13.6	-6.7	-8.9
Hybrid (2+0.125)	Implicit (0+51)	-12.7	-14.0	-17.1	-26.2	-23.4	-18.9	-10.5	-17.5
Hybrid (2+0.125)	Implicit (0+6)	-9.4	-10.7	-13.0	-20.3	-17.6	-12.3	-8.6	-13.1
Hybrid (2+0.125)	Implicit (0+5)	-9.9	-12.0	-12.9	-21.2	-17.7	-14.1	-7.4	-13.6
Hybrid (2+0.125)	Implicit (0+4)	-6.9	-7.7	-11.3	-16.5	-14.8	-2.6	-4.3	-9.2
Hybrid (2+0.125)	Hybrid (3+48)	-17.7	-15.6	-18.8	-26.6	-24.0	-25.4	-15.1	-20.5
Hybrid (2+0.125)	Hybrid (3+3)	-16.6	-14.3	-18.6	-25.4	-23.3	-23.5	-14.7	-19.5
Hybrid (2+0.125)	Hybrid (3+2)	-17.5	-14.6	-18.3	-25.3	-22.8	-26.8	-14.8	-20.0
Hybrid (2+0.125)	Hybrid (3+1)	-13.0	-9.2	-13.2	-19.8	-16.2	-19.2	-9.1	-14.2

Table A3. BD-rate (%) comparison of longer sequence training impact on three buffering strategies for inter-frame coding, using BT.709 for color space conversion. The anchor is the variant employing explicit buffering in both motion and inter-frame coding. The values in parentheses, as in Table 2, indicate the number of full-resolution feature maps buffered for coding one input flow map or frame (explicit + implicit).

Motion	Inter	# Frame	UVG	MCL-JCV	HEVC-B	HEVC-C	HEVC-D	HEVC-E	HEVC-RGB	Average
Hybrid (2.125)	Explicit (3)	5	-12.2	-10.3	-12.4	-15.5	-12.2	-18.7	-11.0	-13.2
		10	-18.5	-15.2	-14.5	-17.6	-14.0	-19.9	-13.2	-16.1
Hybrid (2.125)	Implicit (5)	5	-9.9	-12.0	-12.9	-21.2	-17.7	-14.1	-7.4	-13.6
		10	-20.6	-20.4	-18.3	-25.3	-19.3	-12.8	-12.6	-18.5
Hybrid (2.125)	Hybrid (2)	5	-17.5	-14.6	-18.3	-25.3	-22.8	-26.8	-14.8	-20.0
		10	-24.5	-21.7	-22.5	-29.5	-25.8	-28.8	-20.3	-24.7

Table A4. BD-rate (%) comparison between our HyTIP and the state-of-the-art methods in terms of PSNR-RGB, using BT.709 for color space conversion. The anchor is VTM 17.0. Negative BD-rates suggest bitrate savings.

	UVG	MCL-JCV	HEVC-B	HEVC-C	HEVC-D	HEVC-E	HEVC-RGB	Average
VTM [2]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
HM [1]	26.0	35.7	31.9	29.9	29.8	31.9	29.6	30.7
MaskCRT [9]	6.4	19.1	9.2	29.0	4.7	26.1	-7.2	12.5
DCVC-TCM [35]	35.0	39.3	34.1	62.5	25.5	70.4	19.2	40.9
DCVC-HEM [20]	-6.2	-0.7	-0.4	19.4	-5.3	9.3	-14.2	0.3
DCVC-DC [21]	-24.0	-18.4	-16.4	-11.6	-29.0	-25.4	-32.1	-22.4
DCVC-FM [22]	-24.0	-15.0	-16.5	-14.9	-31.1	-32.0	-23.1	-22.4
HyTIP (Ours)	-22.3	-9.7	-16.3	-6.8	-25.6	-13.9	-27.5	-17.4

Table A5. BD-rate (%) comparison between our HyTIP and the state-of-the-art methods in terms of MS-SSIM-RGB, using BT.709 for color space conversion. The anchor is VTM 17.0. Negative BD-rates suggest bitrate savings.

	UVG	MCL-JCV	HEVC-B	HEVC-C	HEVC-D	HEVC-E	HEVC-RGB	Average
VTM [2]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
HM [1]	21.4	31.5	28.9	29.0	29.7	29.6	26.1	28.0
MaskCRT [9]	-23.0	-30.8	-38.6	-29.0	-43.4	-31.9	-43.8	-34.4
DCVC-TCM [35]	-9.5	-22.5	-26.5	-17.4	-33.8	-18.7	-27.7	-22.3
DCVC-HEM [20]	-28.8	-43.3	-47.7	-40.4	-52.8	-53.1	-46.3	-44.6
DCVC-DC [21]	-36.4	-50.4	-55.7	-51.4	-61.2	-66.0	-56.4	-53.9
HyTIP (Ours)	-38.9	-50.5	-56.4	-50.2	-59.5	-62.6	-58.5	-53.8

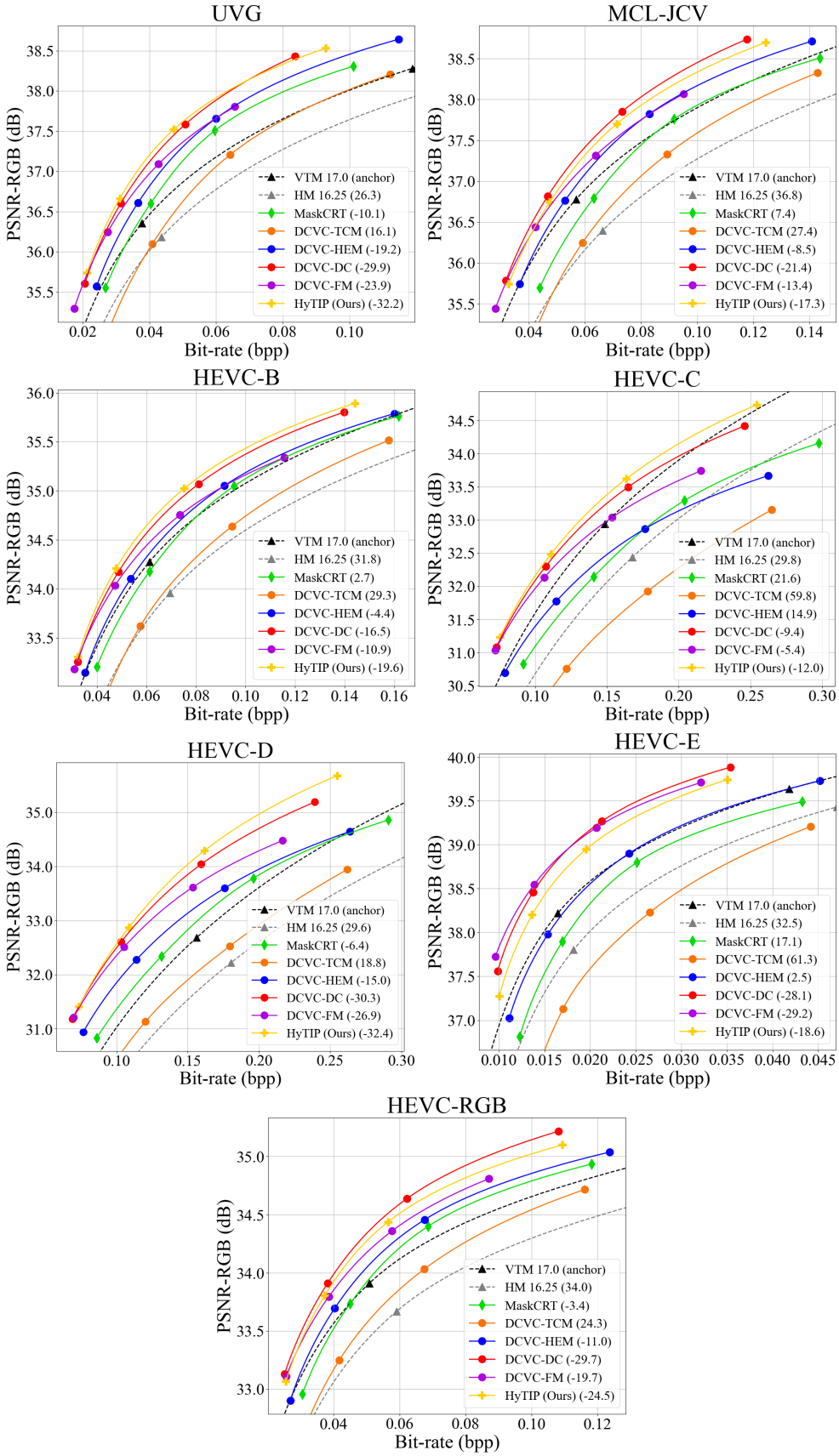


Figure A1. Rate-distortion comparison with state-of-the-art methods in terms of PSNR-RGB, using BT.601 for color space conversion. The values in parentheses represent BD-rates, with VTM 17.0 serving as the anchor.

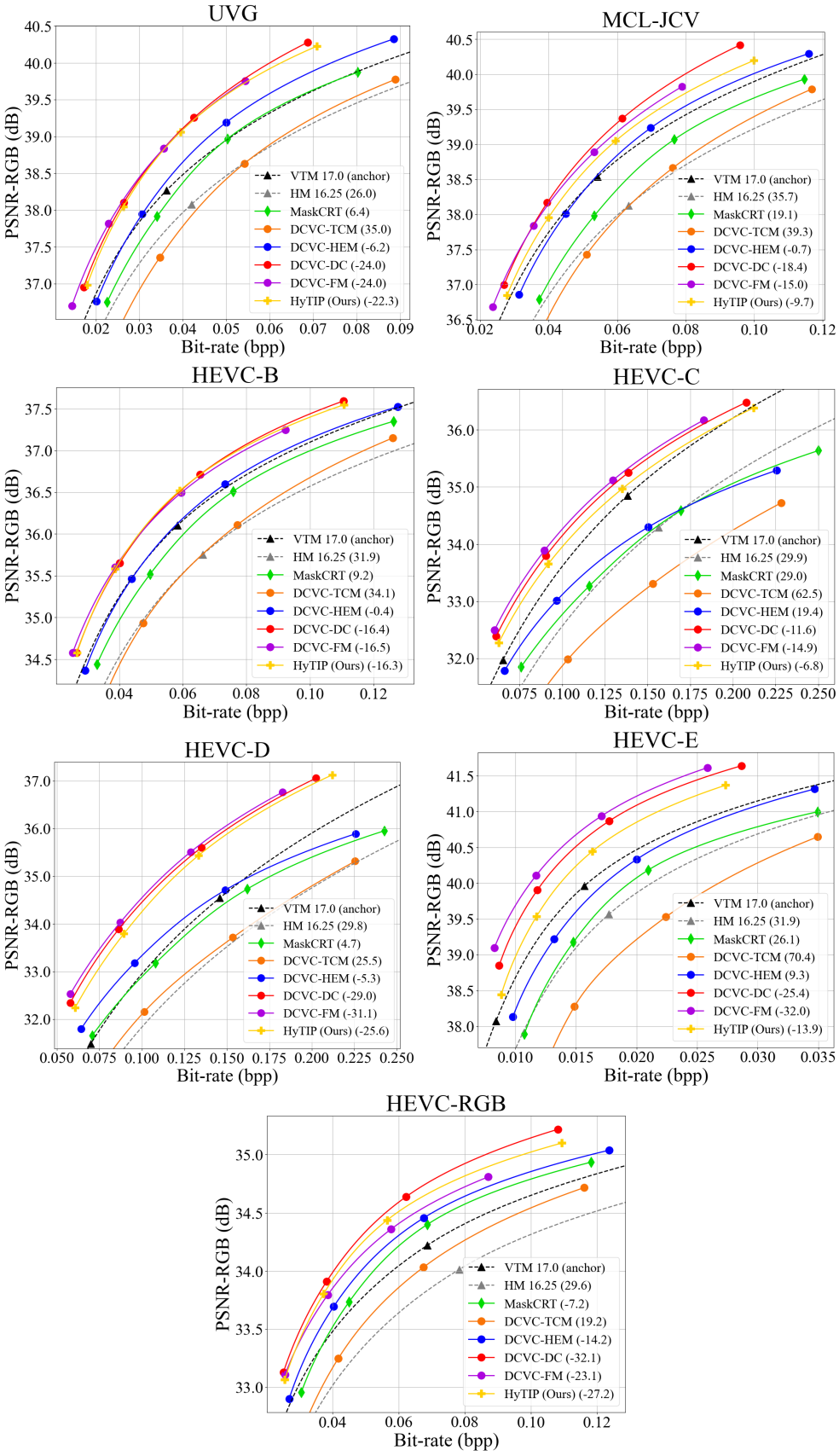


Figure A2. Rate-distortion comparison with state-of-the-art methods in terms of PSNR-RGB, using BT.709 for color space conversion. The values in parentheses represent BD-rates, with VTM 17.0 serving as the anchor.

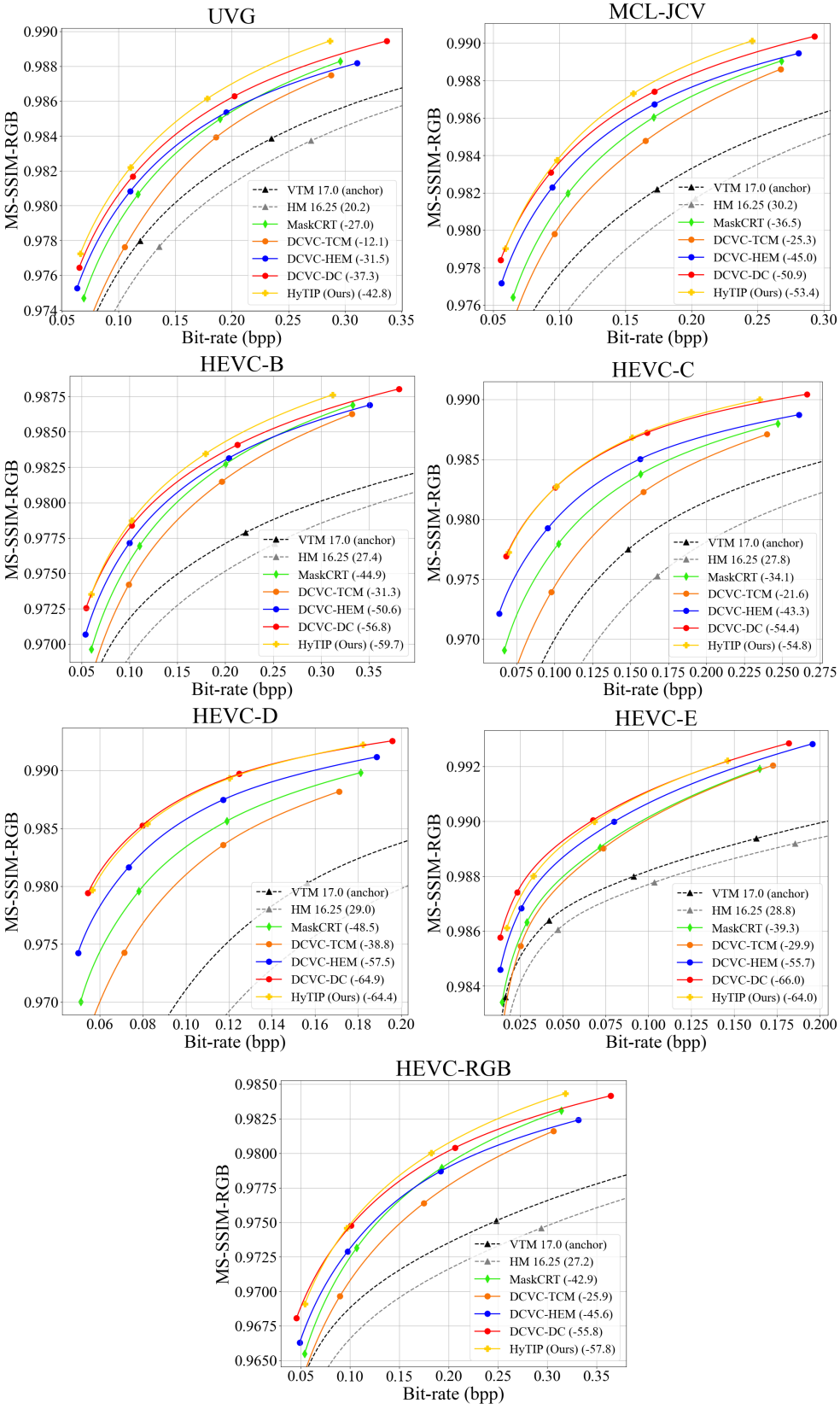


Figure A3. Rate-distortion comparison with state-of-the-art methods in terms of MS-SSIM-RGB, using BT.601 for color space conversion. The values in parentheses represent BD-rates, with VTM 17.0 serving as the anchor.

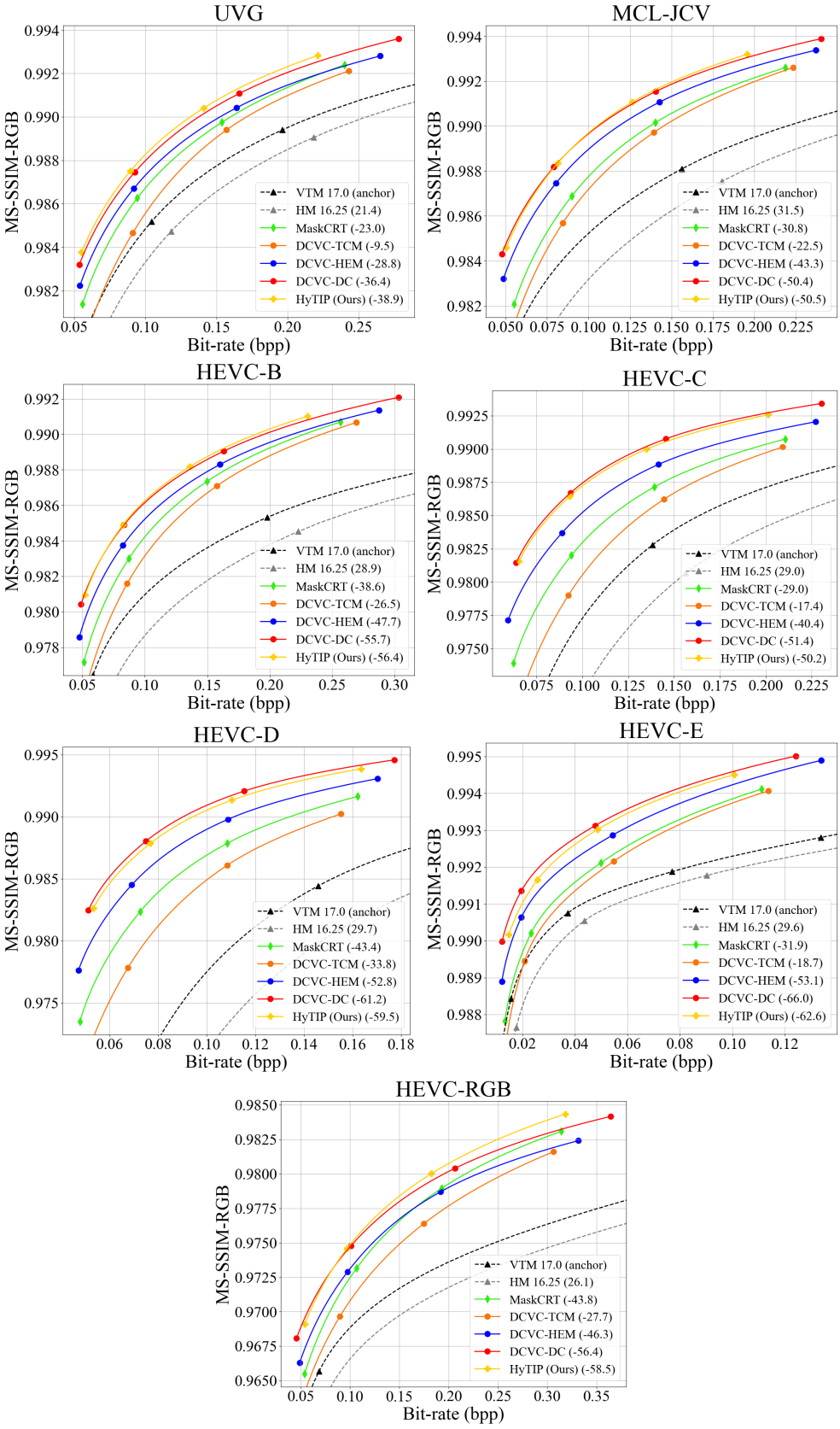


Figure A4. Rate-distortion comparison with state-of-the-art methods in terms of MS-SSIM-RGB, using BT.709 for color space conversion. The values in parentheses represent BD-rates, with VTM 17.0 serving as the anchor.

Table A6. Training procedure. MENet, TCM, FA (frame type adaptation), CTM, and gain units ($s_t^{enc}, s_t^{dec}, s_t^{recon}, s_t^{tcm}$ in Fig. A5, Fig. A7, and Fig. A6) represent the motion estimation network, the temporal context mining module in the inter-frame codec, the hierarchical quality structure [22], the channel transform module [9], and the variable-rate modules, respectively. EPA is the error propagation aware training in [28]. Ref represents the characteristic of reference temporal information in the inter-frame codec.

Phase	# Frames	Training Modules	Loss	lr	Epoch
Motion Coding (Ref: Explicit)	3	Motion codec	$R_t^{motion} + \lambda \times D(x_t, warp(x_{t-1}, \hat{f}_t))$	1e-4	8
Motion Compensation (Ref: Explicit)	3	TCM	$\lambda \times D(x_t, x_c)$	1e-4	10
Inter-frame Coding (Ref: Explicit)	2	Inter-frame codec and Mask Generator	$R_t + \lambda \times D(x_t, \hat{x}_t)$	1e-4	2
Motion Compensation (Ref: Explicit)	3	TCM	$R_t + \lambda \times \frac{D(x_t, x_c) + D(x_t, \hat{x}_t)}{2}$	1e-4	3
Inter-frame Coding (Ref: Explicit)	3 5	All modules except MENet, motion codec, and 3x3 Conv in Fig. 2	$R_t + \lambda \times D(x_t, \hat{x}_t)$ $R_t + \lambda \times D(x_t, \hat{x}_t)$	1e-4 1e-4	8 5
Fine-tuning (Ref: Explicit)	3 5	All modules except MENet and 3x3 Conv in Fig. 2	$R_t + \lambda \times D(x_t, \hat{x}_t)$ $R_t + \lambda \times D(x_t, \hat{x}_t)$	1e-4 1e-4	6 5
Feature Generation (Ref: Hybrid)	3	3x3 Conv in Fig. 2	$R_t + \lambda \times D(x_t, \hat{x}_t)$	1e-4	3
Motion Compensation (Ref: Hybrid)	3	TCM	$R_t + \lambda \times \frac{D(x_t, x_c) + D(x_t, \hat{x}_t)}{2}$	1e-4	4
Inter-frame Coding (Ref: Hybrid)	3 5	All modules except MENet and motion codec	$R_t + \lambda \times D(x_t, \hat{x}_t)$ $R_t + \lambda \times D(x_t, \hat{x}_t)$	1e-4 1e-4	8 4
Fine-tuning (Ref: Hybrid)	3 5	All modules except MENet	$R_t + \lambda \times D(x_t, \hat{x}_t)$ $R_t + \lambda \times D(x_t, \hat{x}_t)$	1e-4 1e-4	3 4
Fine-tuning with EPA (Ref: Hybrid)	5 5	All modules except MENet All modules	$R_t + \lambda \times D(x_t, \hat{x}_t)$ $R_t + \lambda \times D(x_t, \hat{x}_t)$	1e-5 1e-5	4 4
FA (Ref: Hybrid) FA with EPA (Ref: Hybrid)	5 5	Frame Type Conv layers in Fig. A7	$R_t + \lambda \times D(x_t, \hat{x}_t)$ $R_t + \lambda \times D(x_t, \hat{x}_t)$	1e-4 1e-5	1 1
Fine-tuning with EPA (Ref: Hybrid)	5 5	All modules except MENet All modules	$R_t + \lambda \times D(x_t, \hat{x}_t)$ $R_t + \lambda \times D(x_t, \hat{x}_t)$	1e-5 1e-5	1 5
CTM (Ref: Hybrid)	5	CTM in Fig. A6	$R_t + \lambda \times D(x_t, \hat{x}_t)$	1e-4	1
Fine-tuning with EPA (Ref: Hybrid)	5 5	All modules except MENet All modules	$R_t + \lambda \times D(x_t, \hat{x}_t)$ $R_t + \lambda \times D(x_t, \hat{x}_t)$	1e-5 1e-5	2 7
Context Model Training with EPA (Ref: Hybrid)	5 5	Context Model in Fig. A6 Context Model, hyperprior, and decoder in Fig. A6	$R_t + \lambda \times D(x_t, \hat{x}_t)$ $R_t + \lambda \times D(x_t, \hat{x}_t)$	1e-4 1e-4	1 2
Inter-frame Coding with EPA (Ref: Hybrid)	5 5	Inter-frame codec and Mask Generator All modules except MENet and motion codec	$R_t + \lambda \times D(x_t, \hat{x}_t)$ $R_t + \lambda \times D(x_t, \hat{x}_t)$	1e-4 1e-4	3 2
Fine-tuning with EPA (Ref: Hybrid)	5 5	All modules except MENet All modules	$R_t + \lambda \times D(x_t, \hat{x}_t)$ $R_t + \lambda \times D(x_t, \hat{x}_t)$	1e-5 1e-5	3 6
Variable-rate Training with EPA (Ref: Hybrid)	5 5	Gain units in Fig. A5, Fig. A7, and Fig. A6 All modules	$R_t + \lambda \times D(x_t, \hat{x}_t)$ $R_t + \lambda \times D(x_t, \hat{x}_t)$	1e-5 1e-5	8 20
Long-sequence Training with EPA (Ref: Hybrid)	7 10	All modules All modules	$R_t + \lambda \times D(x_t, \hat{x}_t)$ $R_t + \lambda \times D(x_t, \hat{x}_t)$	1e-5 1e-6	1 50

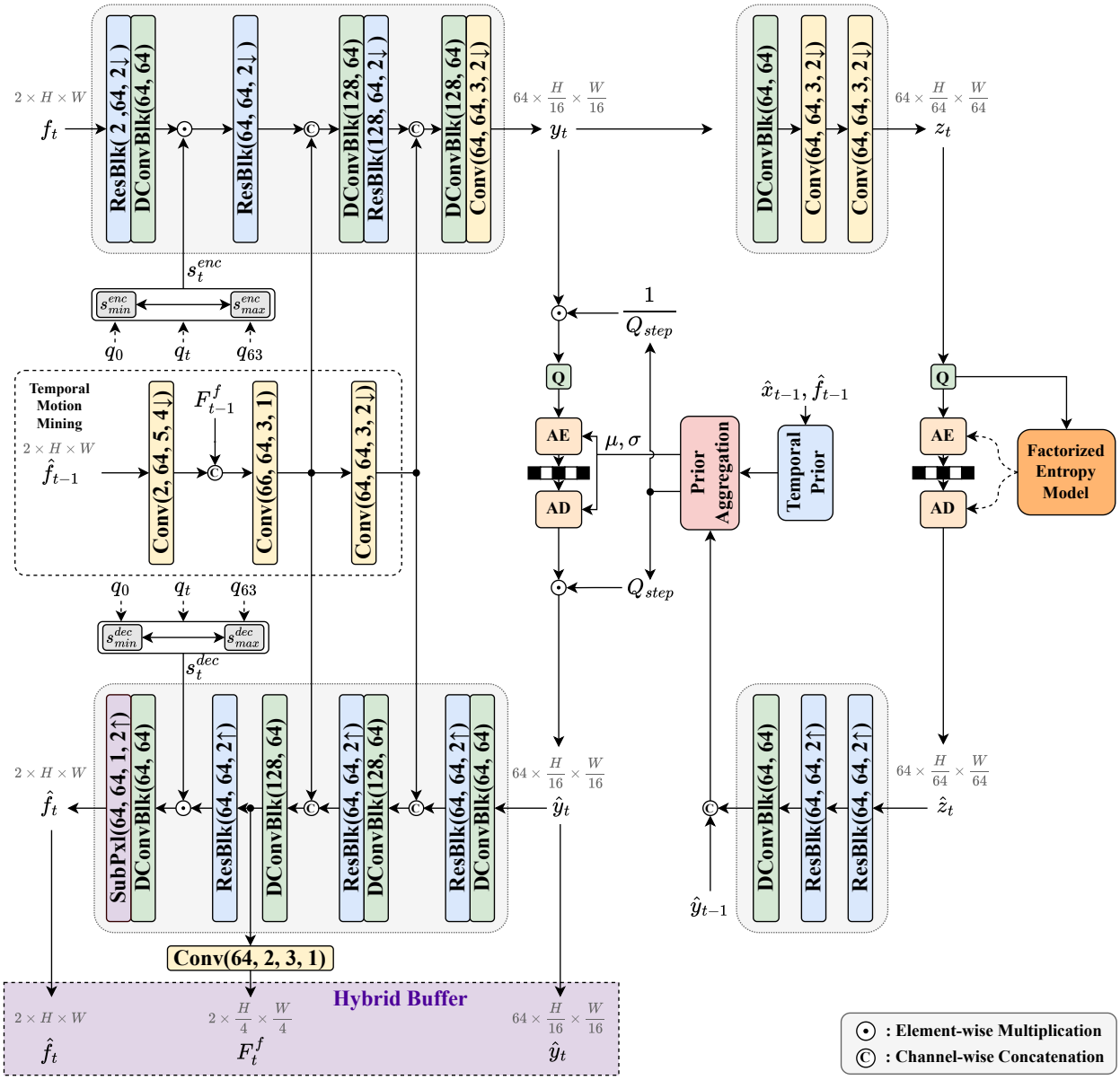


Figure A5. Network architecture detail of our motion codec.

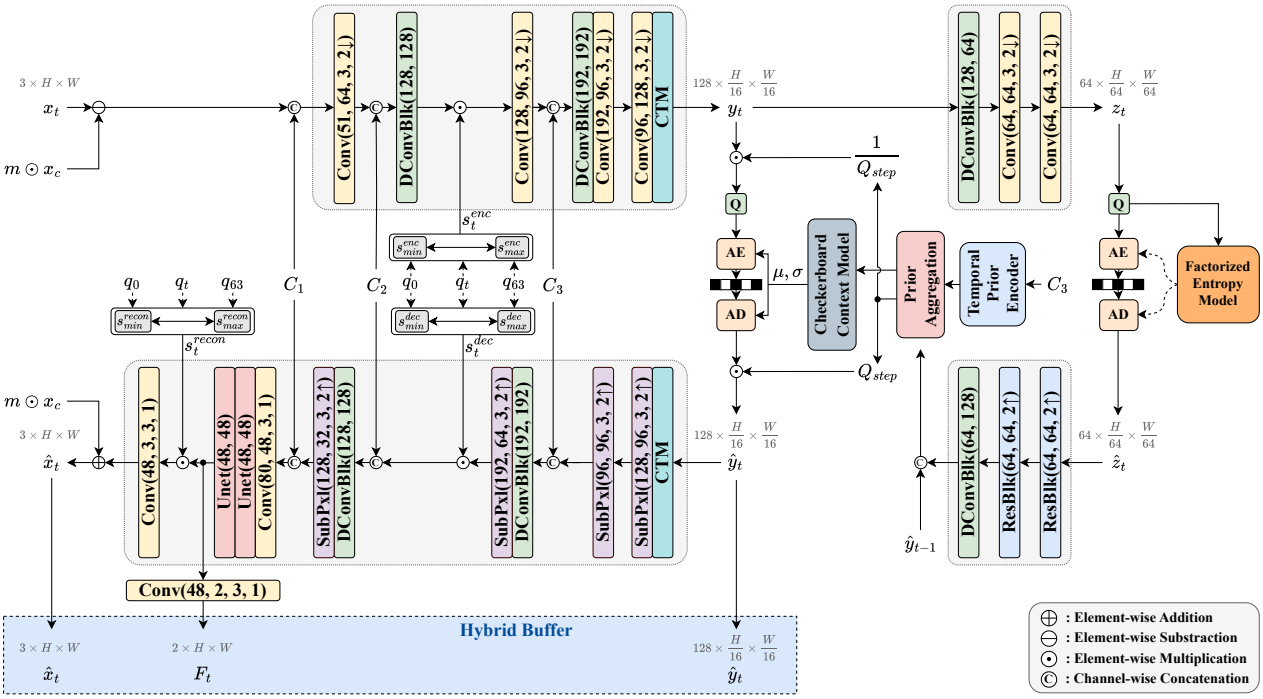


Figure A6. Network architecture detail of our inter-frame codec.

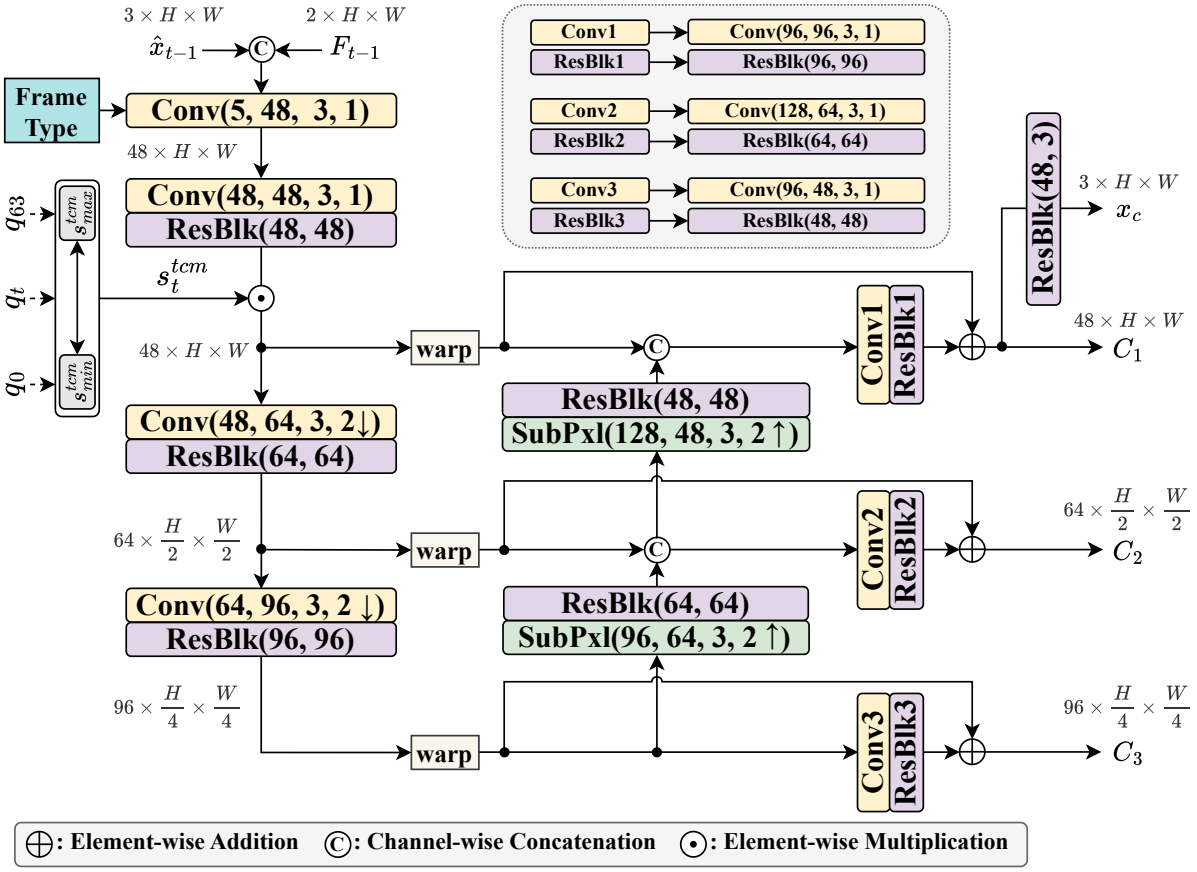


Figure A7. Network architecture detail of our temporal context mining in our inter-frame codec.