EFFORTLESS CROSS-PLATFORM VIDEO CODEC: A CODEBOOK-BASED METHOD

Anonymous authors
Paper under double-blind review

ABSTRACT

Under certain circumstances, advanced neural video codecs can surpass the most complex traditional codecs in their rate-distortion (RD) performance. One of the main reasons for the high performance of existing neural video codecs is the use of the entropy model, which can provide more accurate probability distribution estimations for compressing the latents. This also implies the rigorous requirement that entropy models running on different platforms should use consistent distribution estimations. However, in cross-platform scenarios, entropy models running on different platforms usually yield inconsistent probability distribution estimations due to floating point computation errors that are platform-dependent, which can cause the decoding side to fail in correctly decoding the compressed bitstream sent by the encoding side. In this paper, we propose a cross-platform video compression framework based on codebooks, which avoids autoregressive entropy modeling and achieves video compression by transmitting the index sequence of the codebooks. Moreover, instead of using optical flow for context alignment, we propose to use the conditional cross-attention module to obtain the context between frames. Due to the absence of autoregressive modeling and optical flow alignment, we can design an extremely minimalist framework that can greatly benefit computational efficiency. Importantly, our framework no longer contains any distribution estimation modules for entropy modeling, and thus computations across platforms are not necessarily consistent. Experimental results show that our method can outperform the traditional H.265 (medium) even without any entropy constraints, while achieving the cross-platform property intrinsically.

1 INTRODUCTION

In recent years, neural network-based video codecs have attracted much attention in academia and industry. The rate-distortion (RD) performance of the latest neural video codecs (NVCs) has exceeded that of state-of-the-art traditional video codecs (e.g., H.266/VTM) to some extent (Bross et al., 2021; Wang et al., 2023; Li et al., 2023; 2022). An important reason why these NVCs are able to achieve high performance is the use of the entropy model, which improves the metrics by modeling the temporal redundancy information in the previous frame and the spatial correlation within the current frame, thus reducing the redundancy information in the compression process (Lu et al., 2019; Agustsson et al., 2020; Li et al., 2021a; 2022; 2023; Lin et al., 2020; Rippel et al., 2021; Wang et al., 2023; Zou et al., 2022; Xiang et al., 2022; Yang et al., 2020; Dosovitskiy et al., 2015; Ranjan & Black, 2017; Sun et al., 2018; Hui et al., 2018; Wang et al., 2023; Liu et al., 2022; Yang et al., 2023; Yang & Mandt, 2022; Kwan et al., 2023; Salman Ali et al., 2023).

However, designing a *cross-platform* NVC that can be applied in practice still faces a serious challenge. In cross-platform scenarios, most entropy model-based video codecs face non-deterministic computational problems, such as the incorrect reconstruction in Fig. 1. The non-deterministic computational problem is a common problem caused by floating point operations on different hardware or software platforms, because floating point errors on different platforms obey different distributions. As initially defined by Ballé et al. (2019), the non-determinism problem in cross-platform scenarios cannot be avoided when arithmetic coding is used for data compression. The principle is that different platforms introduce different systematic errors in the entropy modeling (Ballé et al., 2016; 2018; Li et al., 2022; Wang et al., 2023; Li et al., 2023). Specifically, as shown in Fig. 1, we separate the conventional neural compression pipeline into encoder and decoder sides. At the

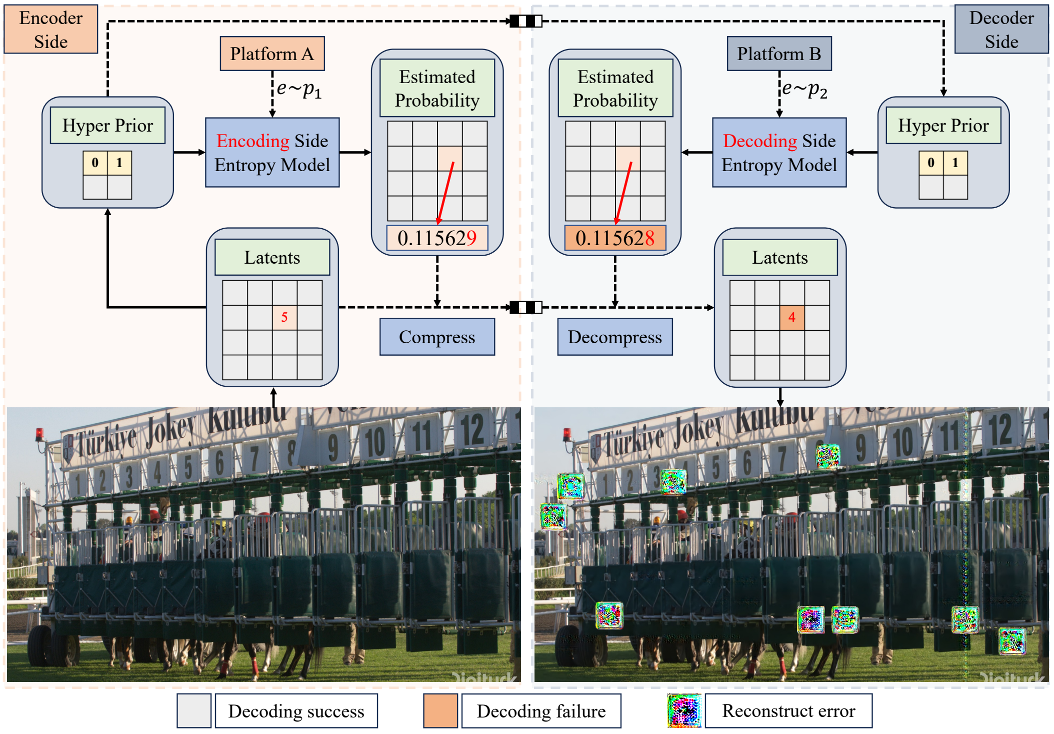

Figure 1: When the encoder and decoder run in cross-platform scenarios, the decoder will reconstruct an incorrect image on account of floating point math.

encoder side, the hyperprior is handled by the entropy model to obtain the estimated distribution of the latents, thus compressing the latents into the bitstream with a high compression ratio. At the encoding end of Platform A, the systematic error e that follows the distribution p_1 is introduced by the entropy model, which will introduce another systematic error $e \sim p_2$ in the decoding end of Platform B. For the same hyperprior, computations at different ends yield slightly different estimates of the probability distribution, which results in the inability to recover the latent from the compressed bitstream completely and correctly using arithmetic decoding. Finally, it will lead to reconstruction errors, as shown in the bottom right image of Fig. 1.

Existing methods address non-deterministic problems mainly through quantization techniques, i.e., replacing uncertain floating point calculations with deterministic integer calculations (Ballé et al., 2019; Sun et al., 2021; Koyuncu et al., 2022; He et al., 2022). Nevertheless, all these methods require data calibration, which makes it complicated to deploy. Moreover, current neural network frameworks (e.g., Pytorch and TensorFlow) cannot support full-int computation, and thus custom implemented modules are necessary. There is also another solution to ensure data consistency between different platforms by transmitting calibration information, which yields a significant performance degradation if the systematic errors across different platforms are very large (Tian et al., 2023).

In this paper, we propose a codebook-based video compression framework by encoding video as index sequences of codebooks. Since no distribution estimation is required, there is no cross-platform problem theoretically. When we receive the correct index sequences on any decoding side, the decoding process is simplified to a codebook-based reconstruction problem.

Specifically, to achieve more effective information compensation, we design different codebooks for intra-frame (i.e., keyframe) and inter-frame (i.e., predicted frame) compression. For keyframes, we rely only on spatial redundancy information for image compression, with the codebook focusing on image reconstruction. For predicted frames, we use the reconstructed latents of the previous frame as the reference. Then, the required bitstream is reduced by another codebook that emphasizes information compensation. In order to better compress the predicted frames, we use a context model based on cross-attention to achieve the integration of latents from multiple frames. The computational complexity of the attention layer is reduced with a window-based strategy. Furthermore, the entropy model-based video compression frameworks accomplish different compression ratios by trading off between rate and distortion. In contrast, we achieve control over different bitrates and distortions by modifying the size of the codebooks.

Our proposed framework has several advantages over existing neural video codecs.

• We have addressed the cross-platform problem completely, as probability distribution estimation is no longer present in our method.

• We propose a cross-attention-based context model for temporal redundancy and spatial redundancy fusing. Due to the absence of autoregressive modeling and optical flow alignment, our proposed framework is extremely minimalist, which can greatly benefit computational efficiency.

2 Related Work

2.1 Cross-Platform Problem in Video Compression

The problem of computational inconsistency of image compression models in cross-platform scenarios was first identified by Ballé et al. (2019). They analyzed the reasons why other insensitive methods cannot avoid this common problem, since most encoding and decoding algorithms use arithmetic coding for data compression (Duda, 2009; Witten et al., 1987; Howard & Vitter, 1994; Cui et al., 2021; Guo et al., 2021). Consequently, to avoid floating point math in cross-platform, they proposed an integer-arithmetic-only network designed for learning-based image compression. A more complex entropy model, which is based on a Gaussian Mixed entropy model (GMM) and context modeling, was quantified by Koyuncu et al. (2022). He et al. (2022) used post-training quantization (PTQ) to train an integer-arithmetic-only model, thus enabling a general quantization technique for image compression. Existing methods are very similar to general model quantification techniques, classified as post-training quantization (PTQ, (Nagel et al., 2019; 2020; Li et al., 2021b)) and quantization-aware training (QAT, (Jacob et al., 2018; Esser et al., 2019; Bhalgat et al., 2020; Krishnamoorthi, 2018; Sun et al., 2021)). Tian et al. (2023) proposed calibration information transmitting (CIT) strategy, which encodes the error-prone entropy parameter coordinates into the auxiliary bitstream, to achieve consistency between the encoder and decoder.

2.2 Neural Video Compression

DVC initially replaces all the components in the traditional hybrid video codec with an end-to-end neural network (Lu et al., 2019). DVC-Pro employs a more efficient network for residual/motion compression and the corresponding refinement network (Lu et al., 2020). To better handle the case of content loss and dramatic motion, Agustsson et al. (2020) proposed a scale-space flow that extends the optical flow-based estimation to 3D transformation. Hu et al. (2020) used multi-resolution instead of single-resolution compression of motion vectors to optimize rate-distortion.

Derived from the residual coding, DCVC employs contextual coding to compensate for the shortcomings of residual coding (Li et al., 2021a). Mentzer et al. (2022) proposed transformer-based temporal models for explicit motion estimation, warping, and residual coding. AlphaVC introduces various techniques to improve the rate-distortion performance, such as conditional I-frame and pixel-to-feature motion prediction (Shi et al., 2022). Li et al. (2022) used multiple modules, such as learnable quantization and parallel entropy model, to greatly improve the performance surpassing the latest VTM codec. MobileCodec is the first-ever inter-frame neural video decoder to run in real-time on a commercial mobile phone, regardless of the cross-platform considerations (Le et al., 2022).

2.3 Vector Quantization

Most of the existing works of Vector Quantization (VQ) focus on image representation and generation. Van Den Oord et al. (2017) proposed the Vector Quantized Variational Autoencoder (VQVAE), a method for learning a discrete representation of an image. Esser et al. (2021) used an encoder-decoder structure to train a quantizer that embeds images into compact sequences using discrete tokens from the learned codebook.

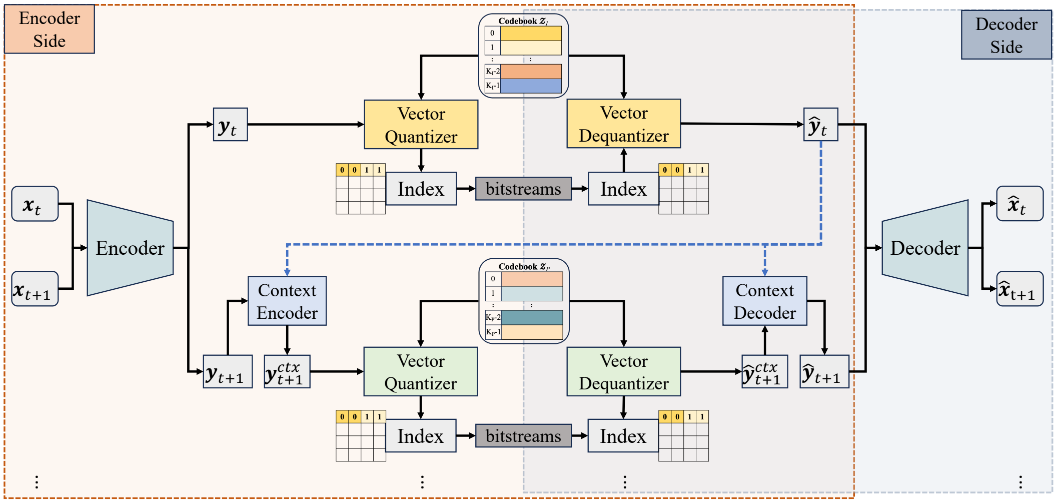

Figure 2: Our method is categorized into keyframes and predicted frames. We propose a cross-attention-based context model for temporal redundancy and spatial redundancy fusing. Due to the absence of autoregressive modeling and optical flow alignment, our proposed framework is minimalist. Keyframes and predicted frames depend on different codebooks for vector quantization.

Some research focuses on the compression of images using VQ. Duan et al. (2023) proposed a hierarchical quantized VAE for coarse-to-fine image compression that narrows the gap between image compression and generation. Kang et al. (2022) obtained the image residual through a three-way autoregressive model and used VQ-VAE as an entropy model for distribution estimation to achieve image compression. Without an entropy model, Zhu et al. (2022) proposed a multi-stage multi-codebook method for image compression.

3 Proposed Method

Our framework is designed accordingly for keyframes $\boldsymbol{x}_t$ and predicted frames $\boldsymbol{x}_{t+1}$, as shown in Fig. 2. Usually, we encode and decode the video in a group of pictures (GOP), the first frame within each GOP is the keyframe, and the other frames are predicted frames.

For keyframes, we use codebook-based methods for image compression (Zhu et al., 2022). While, for the more important predicted frames in video compression, we propose a cross-attention-based context model for temporal redundancy and spatial redundancy fusing, which no longer relies on optical flow alignment and autoregressive modeling. Window-based attention is presented to reduce the computation effort. Specifically, the context encoder integrates the reference latents $\hat{\boldsymbol{y}}_t$ with the current latents $\boldsymbol{y}_{t+1}$ to obtain the context information $\boldsymbol{y}^{ctx}_{t+1}$, while the context decoder recovers the reconstructed latents $\hat{\boldsymbol{y}}_{t+1}$ from the reference latents $\hat{\boldsymbol{y}}_t$ and the quantized context $\hat{\boldsymbol{y}}^{ctx}_{t+1}$.

It is notable that we use the same encoder and decoder in the keyframes and predicted frames to ensure the consistency of the reference latents. Benefiting from the cross-attention interaction, the reference latents and current latents are not necessarily well aligned, and thus registration based on optical flow could be omitted.

3.1 Framework Overview

As shown in Fig. 2, our model contains three primary symmetric components, *encoder* and *decoder*, *context encoder* and *context decoder*, and *vector quantizer* and *vector dequantizer*.

Image encoder and decoder. For an input frame $\boldsymbol{x} \in \mathbb{R}^{H \times W \times 3}$, we obtain the latents $\boldsymbol{y}$ using $\boldsymbol{y} = \mathrm{E}(\boldsymbol{x}) \in \mathbb{R}^{h \times w \times n_c}$, where (H, W) is the image resolution, and (h, w, n_c) represents the dimensions of latents. Then, $\boldsymbol{y}$ is quantized to $\hat{\boldsymbol{y}}$ by vector quantization. For the quantized latents $\hat{\boldsymbol{y}}$, we decode the reconstructed image $\hat{\boldsymbol{x}}$ by using $\hat{\boldsymbol{x}} = \mathrm{D}(\hat{\boldsymbol{y}}) \in \mathbb{R}^{H \times W \times 3}$. $\mathrm{E}(\cdot)$ is the image encoder and $\mathrm{D}(\cdot)$ is the image decoder for both keyframes and predicted frames.

Vector quantization and dequantization. We transform latents $\boldsymbol{y}$ to the codebook index sequence $\boldsymbol{s} \in \mathbb{R}^{h \times w}$, where $s^{ij} \in \{0, ..., K-1\}$ is the index from a learnable codebook $\boldsymbol{\mathcal{Z}} \in \mathbb{R}^{K \times n_c}$ through quantization method, where K is the codebook size, and i, j are the element indices. It performs element-wise quantization $\mathrm{Q}(\cdot)$ of each spatial code $y^{ij} \in \mathbb{R}^{n_c}$ onto the nearest codebook index $k \in \{0, ..., K-1\}$ to obtain the index by

$$s^{ij} = \mathrm{Q}(y^{ij}, \boldsymbol{\mathcal{Z}}) = \arg\min_{k} \left\| y^{ij} - \boldsymbol{\mathcal{Z}}^{k} \right\|_2 . \tag{1}$$

More precisely, the latents $\boldsymbol{y}$ is represented by a sequence of indices $\boldsymbol{s}$ from the codebook $\boldsymbol{\mathcal{Z}}$ by $\mathrm{Q}(\cdot)$, which is denoted as *vector quantizer* in Fig. 2.

Symmetric with the *vector quantizer*, we recover the quantized latents $\hat{\boldsymbol{y}}$ from the sequence of indices $\boldsymbol{s}$ by dequantization method $\mathrm{deQ}(\cdot)$, replacing each index s^{ij} with their corresponding codebook words $\boldsymbol{\mathcal{Z}}^{s^{ij}}$, which is formulated as

$$\hat{y}^{ij} = \mathrm{deQ}(s^{ij}, \boldsymbol{\mathcal{Z}}) = \boldsymbol{\mathcal{Z}}^{s^{ij}} \in \mathbb{R}^{n_c}. \tag{2}$$

Specifically, the sequence of indices $\boldsymbol{s}$ is reconstructed to the quantized latents $\hat{\boldsymbol{y}}$ by $\mathrm{deQ}(\cdot)$, which is shown by *vector dequantizer* in Fig. 2.

Context encoder and decoder. Conditioned by the reference latents $\hat{\boldsymbol{y}}_t$, the current latents $\boldsymbol{y}_{t+1}$ is transformed into a context latents $\boldsymbol{y}^{ctx}_{t+1}$ by the contextual encoder $\mathrm{ctxE}(\cdot)$ as

$$\boldsymbol{y}^{ctx}_{t+1} = \mathrm{ctxE}(\boldsymbol{y}_{t+1}, \hat{\boldsymbol{y}}_t). \tag{3}$$

The current quantized latents $\hat{\boldsymbol{y}}_{t+1}$ is parsed by contextual decoder $\mathrm{ctxD}(\cdot)$, conditioned on the same reference latents $\hat{\boldsymbol{y}}_t$, which is given by

$$\hat{\boldsymbol{y}}_{t+1} = \mathrm{ctxD}(\hat{\boldsymbol{y}}^{ctx}_{t+1}, \hat{\boldsymbol{y}}_{\boldsymbol{t}}). \tag{4}$$

3.2 Cross-attention-based context model

Different from existing methods that use entropy modeling and arithmetic coding to transmit latents, we compress video by transmitting the sequence of indices from the codebook. To make the latents represented by the codebook more informative and efficient, we employ a context model to integrate temporal redundancy (i.e., redundancy between current latents $\boldsymbol{y}_{t+1}$ and reference latents $\hat{\boldsymbol{y}}_t$) and spatial redundancy (i.e., redundancy between the neighborhoods of current latents $\boldsymbol{y}_{t+1}$). The integrated latents are identified as $\boldsymbol{y}^{ctx}_{t+1}$.

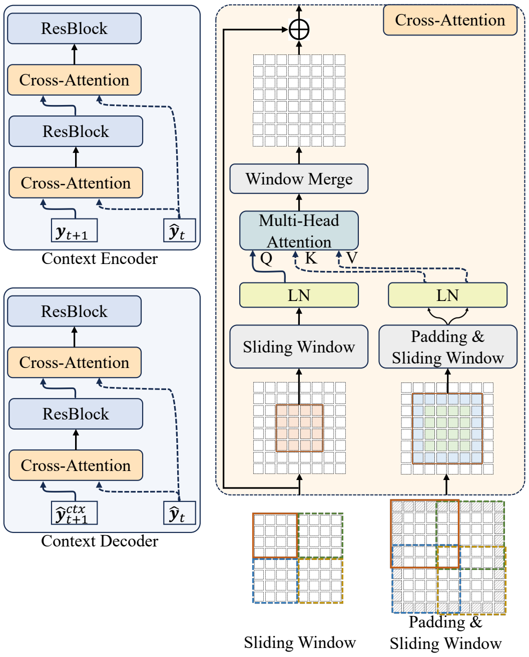

Figure 3: The context model architecture based on cross-attention layer. The structure of the cross-attention layer is described in detail, as well as the *sliding window* and *padding & sliding window* strategies that need to be used in it.

Specifically, as shown in Fig. 3, our context encoder is composed of the cross-attention layer and the resblock layer. Through the cross-attention layer, we encode the temporal redundancy contained in the reference latents $\hat{\boldsymbol{y}}_t$, and then encode the spatial redundancy of the current latents $\boldsymbol{y}_{t+1}$ through the local convolution of the resblock layer. Moreover, this process is repeated twice to achieve more sufficient context fusion.

Similar to the context encoder, the context decoder follows the same structure to combine the quantized context latents $\hat{\boldsymbol{y}}_{t+1}^{ctx}$ and reference latents $\hat{\boldsymbol{y}}_t$ to obtain the quantized latents $\hat{\boldsymbol{y}}_{t+1}$ of the current frame.

Cross-attention (CA). Instead of using optical flow for context alignment, we propose to use the conditional cross-attention module to obtain the context information between frames. Specifically, the cross-attention takes the reference latents as the *key* and *value* while using the current latents as *query*, which no longer requires pixel-level context alignment. Due to the absence of autoregressive modeling and optical flow alignment, our framework is extremely minimalist which can greatly benefit computational efficiency.

Window-based cross-attention (WCA). The global cross-attention of the context model is computationally unaffordable for video compression with high resolution. We use window attention in the context model to alleviate this problem, as visualized in Fig. 3. Specifically, we first partition the current latents into non-overlapping small windows using the *sliding window* method, with a window size of s_{sw}. To encode the motion information between the reference frame and the current frame, we partition the reference latents into partially overlapping windows using a *padding & sliding window* method, with a window size of $s_{psw} = s_{sw} + 2 \times s_p$. Here, s_p is the padding size.

3.3 Multi-Stage Multi-Codebook Vector Quantization

The core problem addressed by classical VQGAN and VQVAE is the discrete representation of an image, which is implemented in the form of the single-stage single-codebook VQ. To obtain high-quality reconstructed images without significantly increasing the size of the codebook and the number of indices that need to be transmitted, Zhu et al. (2022) proposed a multi-stage multi-codebook VQ method. Derived from Zhu et al. (2022), we also employ a multi-stage multi-codebook hierarchy for latent compression of video. Specifically, as shown in Fig. 4, the input latent $\boldsymbol{y}$ is downsampled to obtain the latent $\boldsymbol{y}_d$, which is then quantized by the first multi-codebook vector quantization layer (MCVQ) to obtain the quantized latent $\hat{\boldsymbol{y}}_d$. Since the first MCVQ layer focuses on reconstructing the low-frequency signals, we use the next downsample and MCVQ layer to quantize the high-frequency residual $\boldsymbol{y}'$ between $\boldsymbol{y}_d$ and $\hat{\boldsymbol{y}}_d$ to obtain $\hat{\boldsymbol{y}}_d'$. We use the combination of downsampling and MCVQ three times to achieve progressive compensation from low frequency to high frequency. In addition, the low-frequency signals in an image require more bitstreams to encode than the high-frequency signals (Duan et al., 2023). Based on this principle, we use downsampling between different stages to achieve the spatial dimensionality decrease from low to high frequencies to reduce the number of indices, and gradually reduce the size of the codebook at each stage to minimize the bitlength of a single index.

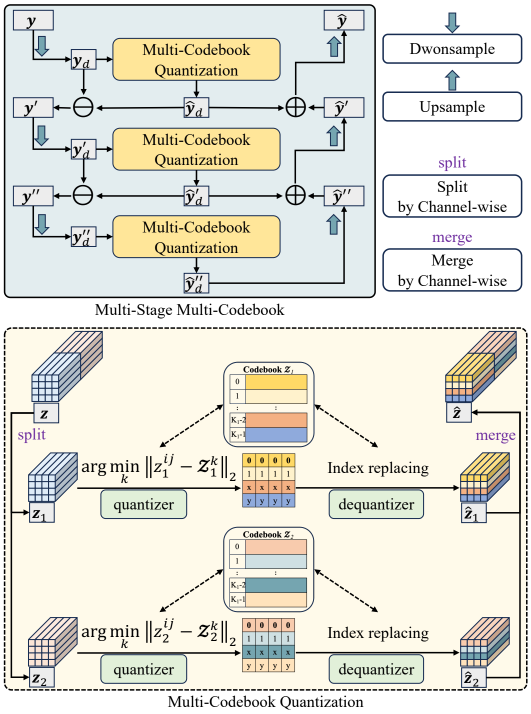

Figure 4: The multi-stage multi-codebook VQ architecture improves the reconstruction capability through successive compensation of codebooks at each stage. Multiple codebooks enhance the VQ representation at each stage.

While decoding, we reconstruct the quantized latents $\hat{\boldsymbol{y}}$ from high-frequency latents $\hat{\boldsymbol{y}}_d''$ to low-frequency latents $\hat{\boldsymbol{y}}_d$ by progressively upsampling the quantized latents of the next level (e.g., $\hat{\boldsymbol{y}}_d''$) and compensating the quantized latents of the previous level by addition (e.g., $\hat{\boldsymbol{y}}'' + \hat{\boldsymbol{y}}_d'$).

For the MCVQ layer, as shown at the bottom of Fig. 4, we first *split* the input latents $\boldsymbol{z}$ in channel dimension to get $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$. Then we quantize and inverse quantize $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ using codebooks $\boldsymbol{\mathcal{Z}}_1$ and $\boldsymbol{\mathcal{Z}}_2$ to obtain $\hat{\boldsymbol{z}}_1$ and $\hat{\boldsymbol{z}}_2$, respectively. Finally, $\hat{\boldsymbol{z}}_1$ and $\hat{\boldsymbol{z}}_2$ are combined in the channel dimension by *merge* to obtain the quantized latents $\hat{\boldsymbol{z}}$.

3.4 Implementation Details

Image encoder and decoder. For both keyframes and predicted frames, we use the same image encoder $\mathrm{E}(\cdot)$ and image decoder $\mathrm{D}(\cdot)$ to transform between image $\boldsymbol{x} \in \mathbb{R}^{H \times W \times 3}$ and latents $\boldsymbol{y} \in \mathbb{R}^{h \times w \times n_c}$. In our experiments, we work with $h = \frac{H}{8}$, $w = \frac{W}{8}$, and $n_c = 128$.

Window-based cross-attention. Instead of using optical flow for context alignment, we propose to use the conditional cross-attention module to obtain the context information between frames. Finally, window-based cross-attention is used to reduce the computational effort, since attention is only considered within local windows. We set the sliding window size s_{sw} of *query* to 4, and the padding size s_p to 2. Then the sliding window size s_{psw} of *key* and *value* will be $4 + 2 \times 2 = 8$.

Codebook settings. As shown in Fig. 4, the multi-stage multi-codebook vector quantization structure that we use in both keyframes and predicted frames employs a three-stage MCVQ and uses two codebooks during every stage of MCVQ.

Existing methods typically achieve different compression ratios by trading off loss weights between bitrate and distortion (Xiang et al., 2022; Li et al., 2022; 2021a; Lu et al., 2019). In contrast, we achieve different bitrates by modulating the codebook sizes used in the predicted frames. Specifically, we use a group of three parameters to specify different codebook sizes in multi-stage multi-codebook vector quantization. The codebook sizes for keyframes are specified as $\{8192, 2048, 512\}$, which are maintained constant at different compression ratios. While for predicted frames, we use three different groups of codebook sizes to achieve different video compression ratios as $\{8192, 2048, 512\}$, $\{64, 2048, 512\}$ and $\{8, 2048, 512\}$.

Optimization Loss. Distortion loss $\mathcal{L}$ is optimized throughout the training process, where $d(\cdot)$ represents the mean square error or MS-SSIM; m is the GOP size used for training, where all frames are treated as predicted frames except the first frame, which is treated as a keyframe.

$$\mathcal{L} = \sum_{t=0}^{m} \underbrace{d(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t)}_{\text{distortion}} \tag{5}$$

4 Experiments

4.1 Experimental Setup

Datasets. We use Vimeo-90k dataset (Xue et al., 2019) for training, which contains 89800 video clips with the resolution of 448×256. The videos are randomly cropped into 256×256 patches. The training GOP size m is set to 7 for each clip of Vimeo-90k containing 7 frames. We evaluate our model using UVG (Mercat et al., 2020), HEVC Class B, and MCL-JCV (Wang et al., 2016) datasets, which both have a resolution of 1920×1080 (1080P). The images are cropped to 1920×1024 by center cropping to ensure that the input image shape is divisible by 128.

Baselines. Neural network-based methods commonly rely on engineering optimizations to replace some modules with integer networks to achieve cross-platform capability. There are also methods that achieve cross-platform consistency by transferring calibration information through plug-ins. Conventional video codecs, including H.264, and H.265, are the only video codecs that can be practically used cross-platform. Different from other neural network-based methods, our algorithm is designed to avoid cross-platform problems. Therefore the conventional codecs H.264 and H.265 in FFmpeg are used as the anchor with *medium* preset.

Metric. We evaluate the performance of all models using common metrics including PSNR and MS-SSIM (Wang et al., 2004).

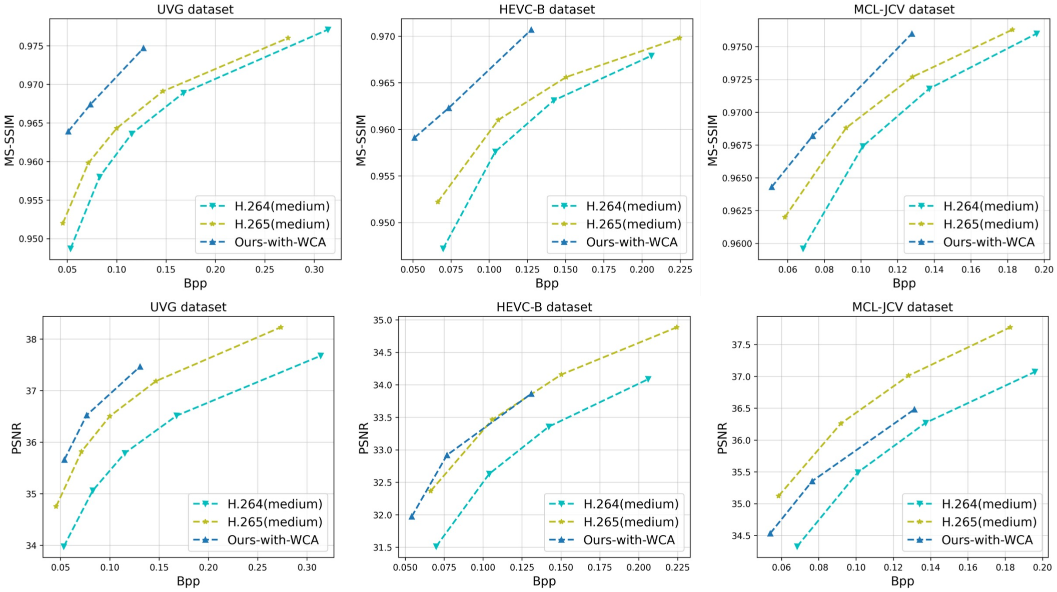

Figure 5: Rate-distortion performance on UVG, HEVC-B, and MCL-JCV datasets.

Table 1: BD-rate calculated by *SSIM* and *PSNR* on test datasets with the anchor H.265 (medium).

	SSIM-BPP			PSNR-BPP		
	H.264	H.265	Ours-w-WCA	H.264	H.265	Ours-w-WCA
UVG	+21.2 %	0 %	-43.7 %	+63.7 %	0 %	-23.7 %
HEVC-B	+18.4 %	0 %	-38.2 %	+41.5 %	0 %	-5.2 %
MCL-JCV	+19.9 %	0 %	-19.1 %	+49.3 %	0 %	+23.7 %
Average	+19.8 %	0 %	-33.7 %	+51.5 %	0 %	-1.7 %

Test conditions. We test each video for 96 frames with GOP size 12 for H.264 and H.265, same in (Tian et al., 2023; Li et al., 2021a). For our model, we use GOP size 32 for testing. The experiments are conducted in cross-platform scenarios, where the videos are encoded with an NVIDIA Tesla V100 machine and decoded with an NVIDIA Tesla P40 machine. Since the existing neural video codecs cannot achieve cross-platform decoding directly, we only compare our method with the conventional codecs.

4.2 Results

Rate-distortion performance. In Fig. 5, we plot the rate-distortion of our method and the baseline methods. Our method improves significantly over H.265 across different datasets in terms of SSIM. For PSNR, we outperform H.265 on UVG and HEVC-B datasets, but not on MCL-JCV. To analyze the reasons for the performance degradation on MCL-JCV, we show one of the videos with little variation and high redundancy *in the Appendix, Section B*. For videos with high redundancy, our method uses a fixed number of indices for video compression which results in performance degradation.

As shown in Table 1, we evaluate quantitative metric with BD-rate (Bjontegaard, 2001) computed from PSNR-BPP and SSIM-BPP, respectively. Our method achieves an average of 33.7% bitrate saving in terms of SSIM, while the improvement on PSNR is limited, with an average of 1.7% bitrate saving compared to H.265. Notably, our method achieves 23.7% bitrate saving on the UVG dataset with evident motion.

Cross-platform results. To verify the cross-platform capability, we encode the video at V100 using our model. On the decoding end, we compare the decoding results obtained from V100 and P40, respectively. We calculate the BD-rate for both decoders when decoding the bitstreams

Table 2: BD-rate calculated on different platforms by *SSIM* and *PSNR* on UVG dataset.

	SSIM-BPP		PSNR-BPP	
Encoder	V100	V100	V100	V100
Decoder	V100	P40	V100	P40
UVG	0 %	0 %	0 %	0 %

Table 3: Model complexity and BD-rate. All models are tested with 1080P video on V100.

			Context	Encoding	Decoding	SSIM	PSNR
	Params	MACs	time	time	time	BD-rate	BD-rate
Resblock-based	53.438M	2.221T	3.2ms	102.2ms	219.5ms	0%	0%
CA-based-64	53.408M	2.219T	74.2ms	242.9ms	290.2ms	-40.8%	-48.0%
WCA-based-4	53.408M	2.233T	**13.0ms**	122.4ms	**229.3ms**	-40.7 %	-49.0 %
light-decoder	46.162M	0.740T	13.1ms	122.2ms	**35.8ms**	-23.7 %	-37.4 %

that are originally encoded by V100, which is shown in Table 2. These error-free results are a solid demonstration of the cross-platform capabilities of our method. *We also conduct experiments on the decoding side running at more platforms, which is shown in the Appendix, Section C.*

4.3 Ablation Studies

Window-based cross-attention. Since the CA-based model will lead to out-of-memory, we use the *WCA-based* model with window size 64 as an alternative comparison option. We implement *Resblock-based*, *CA-based-64*, and *WCA-based-4* models, where *CA-based-64* is the WCA method with window size 64 and *WCA-based-4* is our proposed model. Experimental results in the last two columns of Table 3 show that our proposed *WCA-based* method can deliver bitrate saving of approximately 40% over the *Resblock-based* model in terms of SSIM. *More visualization results are provided in the Appendix, Section D.*

Decoding efficiency. Table 3 shows the complexity comparison of the models in the number of parameters, MACs (multiply-accumulate operations), context model time, encoding time, and decoding time (including context model time). All our models are tested on a server with an NVIDIA Tesla V100 GPU. Our proposed method saves 82.5% of context modeling time when similar performance is obtained using the *WCA-based* methods with different window sizes. It is worth noting that our *light-decoder* model, with fewer parameters, can decode 1080P video in real-time on V100 and still outperform H.265, as demonstrated in the last row of Table 3. *We provide further visualization in the Appendix, Section D and Section E.*

5 Conclusions

In this work, we present a video compression framework based on codebooks. Our method models temporal and spatial redundancy using a *WCA-based* context model, which avoids autoregressive modeling and optical flow alignment. The absence of the entropy model for probability distribution estimation makes our approach inherently effective across platforms, which is crucial for practical applications of neural video codecs. Our method outperforms H.265 in terms of SSIM on multiple datasets, and achieves better PSNR results than H.265 on the UVG and HEVC-B datasets. Our work provides valuable insights into the development of neural video codecs.

Our proposed method for compressing videos to a sequence of indices of the codebook has some limitations. Specifically, our method does not currently impose any constraints on the entropy, which can result in the model delivering constant bitstreams for different videos. As a result, when compressing videos with high redundancy, *as demonstrated in the Appendix, Section B*, our method may perform poorly. Consequently, we will consider designing entropy constraints for the indices of different videos to achieve content-based adaptive compression in future work.

REFERENCES

Eirikur Agustsson, David Minnen, Nick Johnston, Johannes Balle, Sung Jin Hwang, and George Toderici. Scale-space flow for end-to-end optimized video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8503–8512, 2020.

Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D Fairchild. Flip: A difference evaluator for alternating images. *Proc. ACM Comput. Graph. Interact. Tech.*, 3(2):15–1, 2020.

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016.

Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018.

Johannes Ballé, Nick Johnston, and David Minnen. Integer networks for data compression with latent-variable models. In *International Conference on Learning Representations*, 2019.

Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 696–697, 2020.

Gisle Bjontegaard. Calculation of average psnr differences between rd-curves. *ITU SG16 Doc. VCEG-M33*, 2001.

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*, 31(10):3736–3764, 2021.

Ze Cui, Jing Wang, Shangyin Gao, Tiansheng Guo, Yihui Feng, and Bo Bai. Asymmetric gained deep image compression with continuous rate adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10532–10541, 2021.

Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2758–2766, 2015.

Zhihao Duan, Ming Lu, Zhan Ma, and Fengqing Zhu. Lossy image compression with quantized hierarchical vaes. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 198–207, 2023.

Jarek Duda. Asymmetric numeral systems. *arXiv preprint arXiv:0902.0271*, 2009.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883, 2021.

Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019.

Zongyu Guo, Runsen Feng, Zhizheng Zhang, Xin Jin, and Zhibo Chen. Learning cross-scale prediction for efficient neural video compression. *arXiv preprint arXiv:2112.13309*, 2021.

Dailan He, Ziming Yang, Yuan Chen, Qi Zhang, Hongwei Qin, and Yan Wang. Post-training quantization for cross-platform learned image compression. *arXiv preprint arXiv:2202.07513*, 2022.

Paul G Howard and Jeffrey Scott Vitter. Arithmetic coding for data compression. *Proceedings of the IEEE*, 82(6):857–865, 1994.

Zhihao Hu, Zhenghao Chen, Dong Xu, Guo Lu, Wanli Ouyang, and Shuhang Gu. Improving deep video compression by resolution-adaptive flow coding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pp. 193–209. Springer, 2020.

Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8981–8989, 2018.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2704–2713, 2018.

Ning Kang, Shanzhao Qiu, Shifeng Zhang, Zhenguo Li, and Shu-Tao Xia. Pilc: Practical image lossless compression with an end-to-end gpu oriented neural framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3739–3748, 2022.

Esin Koyuncu, Timofey Solovyev, Elena Alshina, and André Kaup. Device interoperability for learned image compression with weights and activations quantization. In *2022 Picture Coding Symposium (PCS)*, pp. 151–155. IEEE, 2022.

Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

Ho Man Kwan, Ge Gao, Fan Zhang, Andrew Gower, and David Bull. Hinerv: Video compression with hierarchical encoding based neural representation. *arXiv preprint arXiv:2306.09818*, 2023.

Hoang Le, Liang Zhang, Amir Said, Guillaume Sautiere, Yang Yang, Pranav Shrestha, Fei Yin, Reza Pourreza, and Auke Wiggers. Mobilecodec: neural inter-frame video compression on mobile devices. In *Proceedings of the 13th ACM Multimedia Systems Conference*, pp. 324–330, 2022.

Jiahao Li, Bin Li, and Yan Lu. Deep contextual video compression. *Advances in Neural Information Processing Systems*, 34:18114–18125, 2021a.

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*, pp. 1503–1511, 2022.

Jiahao Li, Bin Li, and Yan Lu. Neural video compression with diverse contexts. *arXiv preprint arXiv:2302.14402*, 2023.

Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*, 2021b.

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*, pp. 3546–3554, 2020.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986, 2022.

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*, pp. 11006–11015, 2019.

Guo Lu, Xiaoyun Zhang, Wanli Ouyang, Li Chen, Zhiyong Gao, and Dong Xu. An end-to-end learning framework for video compression. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3292–3308, 2020.

Fabian Mentzer, George Toderici, David Minnen, Sung-Jin Hwang, Sergi Caelles, Mario Lucic, and Eirikur Agustsson. Vct: A video compression transformer. *arXiv preprint arXiv:2206.07307*, 2022.

Alexandre Mercat, Marko Viitanen, and Jarno Vanne. Uvg dataset: 50/120fps 4k sequences for video codec analysis and development. In *Proceedings of the 11th ACM Multimedia Systems Conference*, pp. 297–302, 2020.

Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1325–1334, 2019.

Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pp. 7197–7206. PMLR, 2020.

Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4161–4170, 2017.

Oren Rippel, Alexander G Anderson, Kedar Tatwawadi, Sanjay Nair, Craig Lytle, and Lubomir Bourdev. Elf-vc: Efficient learned flexible-rate video coding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14479–14488, 2021.

Muhammad Salman Ali, Yeongwoong Kim, et al. Towards efficient image compression without autoregressive models. *Advances in Neural Information Processing Systems*, 2023.

Xihua Sheng, Jiahao Li, Bin Li, Li Li, Dong Liu, and Yan Lu. Temporal context mining for learned video compression. *IEEE Transactions on Multimedia*, 2022.

Yibo Shi, Yunying Ge, Jing Wang, and Jue Mao. Alphavc: High-performance and efficient learned video compression. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XIX*, pp. 616–631. Springer, 2022.

Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8934–8943, 2018.

Heming Sun, Lu Yu, and Jiro Katto. Learned image compression with fixed-point arithmetic. In *2021 Picture Coding Symposium (PCS)*, pp. 1–5. IEEE, 2021.

Kuan Tian, Yonghang Guan, Jinxi Xiang, Jun Zhang, Xiao Han, and Wei Yang. Towards real-time neural video codec for cross-platform application using calibration information. *arXiv preprint arXiv:2309.11276*, 2023.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Guo-Hua Wang, Jiahao Li, Bin Li, and Yan Lu. Evc: Towards real-time neural image compression with mask decay. *arXiv preprint arXiv:2302.05071*, 2023.

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 *2016 IEEE international conference on image processing (ICIP)*, pp. 1509–1513. IEEE, 2016.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Ian H Witten, Radford M Neal, and John G Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

Jinxi Xiang, Kuan Tian, and Jun Zhang. Mimt: Masked image modeling transformer for video compression. In *The Eleventh International Conference on Learning Representations*, 2022.

Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127:1106–1125, 2019.

Ren Yang, Fabian Mentzer, Luc Van Gool, and Radu Timofte. Learning for video compression with hierarchical quality and recurrent enhancement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6628–6637, 2020.

Ruihan Yang and Stephan Mandt. Lossy image compression with conditional diffusion models. *arXiv preprint arXiv:2209.06950*, 2022.

Ruihan Yang, Yibo Yang, Joseph Marino, and Stephan Mandt. Insights from generative modeling for neural video compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

Xiaosu Zhu, Jingkuan Song, Lianli Gao, Feng Zheng, and Heng Tao Shen. Unified multivariate gaussian mixture for efficient neural image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17612–17621, 2022.

Renjie Zou, Chunfeng Song, and Zhaoxiang Zhang. The devil is in the details: Window-based attention for image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17492–17501, 2022.

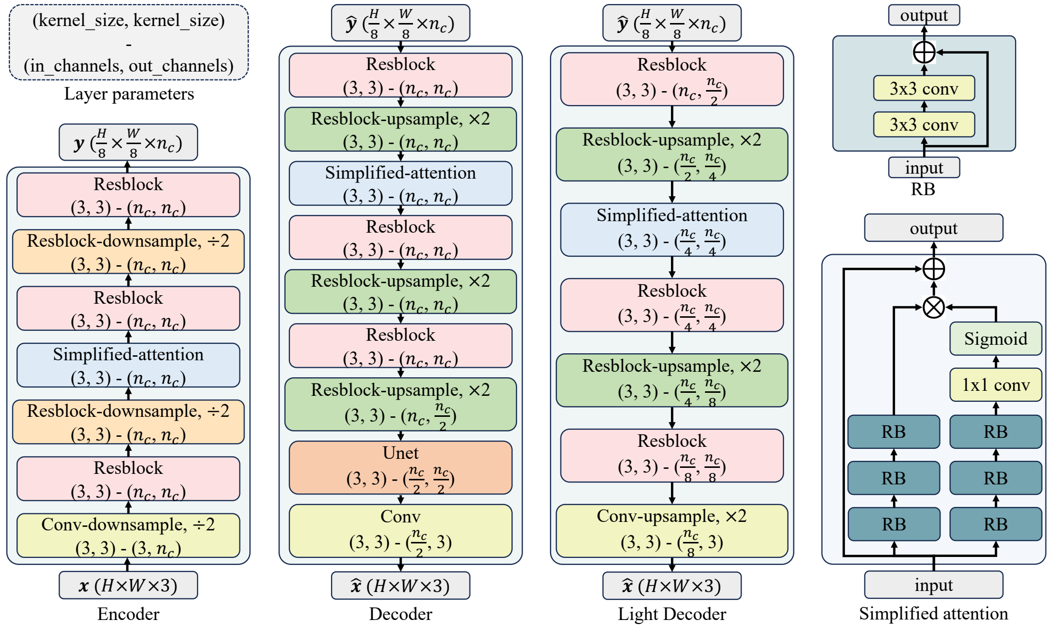
(kernel_size, kernel_size)
-
(in_channels, out_channels)
Layer parameters
y (H/8×W/8×nc)
Resblock (3, 3) - (nc, nc)
Resblock-downsample, ÷2 (3, 3) - (nc, nc)
Resblock (3, 3) - (nc, nc)
Simplified-attention (3, 3) - (nc, nc)
Resblock-downsample, ÷2 (3, 3) - (nc, nc)
Resblock (3, 3) - (nc, nc)
Conv-downsample, ÷2 (3, 3) - (3, nc)
x (H×W×3)
Encoder
ŷ (H/8×W/8×nc)
Resblock (3, 3) - (nc, nc)
Resblock-upsample, ×2 (3, 3) - (nc, nc)
Simplified-attention (3, 3) - (nc, nc)
Resblock (3, 3) - (nc, nc)
Resblock-upsample, ×2 (3, 3) - (nc, nc)
Resblock (3, 3) - (nc, nc)
Resblock-upsample, ×2 (3, 3) - (nc, nc/2)
Unet (3, 3) - (nc/2, nc/2)
Conv (3, 3) - (nc/2, 3)
x̂ (H×W×3)
Decoder
ŷ (H/8×W/8×nc)
Resblock (3, 3) - (nc, nc/2)
Resblock-upsample, ×2 (3, 3) - (nc/2, nc/4)
Simplified-attention (3, 3) - (nc/4, nc/4)
Resblock (3, 3) - (nc/4, nc/4)
Resblock-upsample, ×2 (3, 3) - (nc/4, nc/8)
Resblock (3, 3) - (nc/8, nc/8)
Conv-upsample, ×2 (3, 3) - (nc/8, 3)
x̂ (H×W×3)
Light Decoder
output
3x3 conv
3x3 conv
input
RB
output
Sigmoid
1x1 conv
RB
RB
RB
RB
RB
RB
input
Simplified attention

Figure 6: Image encoder/decoder architecture.

Table 4: Comparison of BPP for the same video in the same reconstructed PSNR condition.

videoSRC30	H.265	Ours-with-WCA
PSNR	37.589	37.483
BPP	0.033	0.054

A Network

Image encoder and decoder. For both keyframes and predicted frames, we use the same image encoder $\mathrm{E}(\cdot)$ and image decoder $\mathrm{D}(\cdot)$ to transform between image $\boldsymbol{x} \in \mathbb{R}^{H\times W\times 3}$ and latents $\boldsymbol{y} \in \mathbb{R}^{h\times w\times n_c}$. The network architectures are shown in Fig. 6. In addition, we show the model architecture of *light decoder*.

Context encoder and decoder. Instead of using optical flow for context alignment, we propose to use the conditional cross-attention module to obtain the context information between frames. Specifically, for more efficient computation and sufficient context information fusing, we design the *WCA-based* context model with the architecture shown in Fig. 7. The architecture of the WCA layer is shown in Fig. 3. The number of *heads* is 8, and the *dimension of each head* is 16.

Vector quantization and dequantization. Derived from Zhu et al. (2022), we also employ a multi-stage multi-codebook hierarchy to vector quantization of the latent. More details about the architecture of multi-stage multi-codebook VQ can be found in (Zhu et al., 2022).

B Visualization of sample video of MCL-JCV

As shown in Figure. 8, we select a typical video sequence in the MCL-JCV dataset, in which the small target (i.e., the bird in the video) undergoes a slight motion. Our method uses a fixed number of indices for video compression, which inevitably wastes indices in videos with high redundancy, resulting in higher bitrates for the same reconstruction quality, as shown in Table 4.

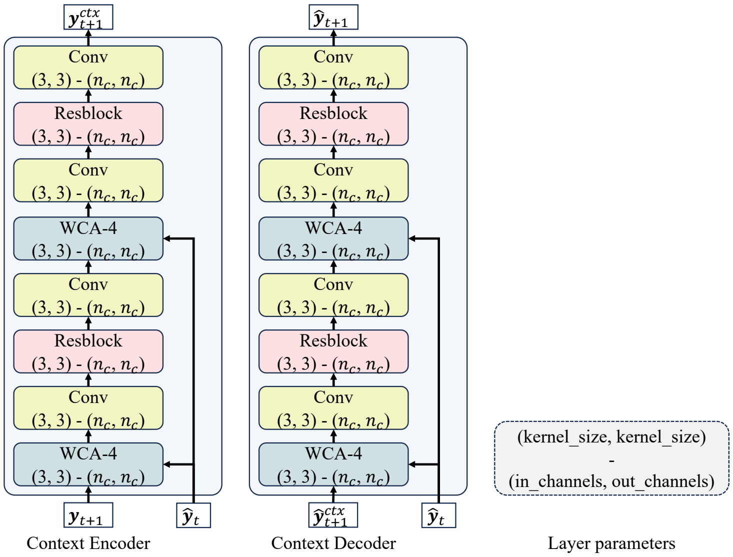

Figure 7: Context encoder/decoder architecture.

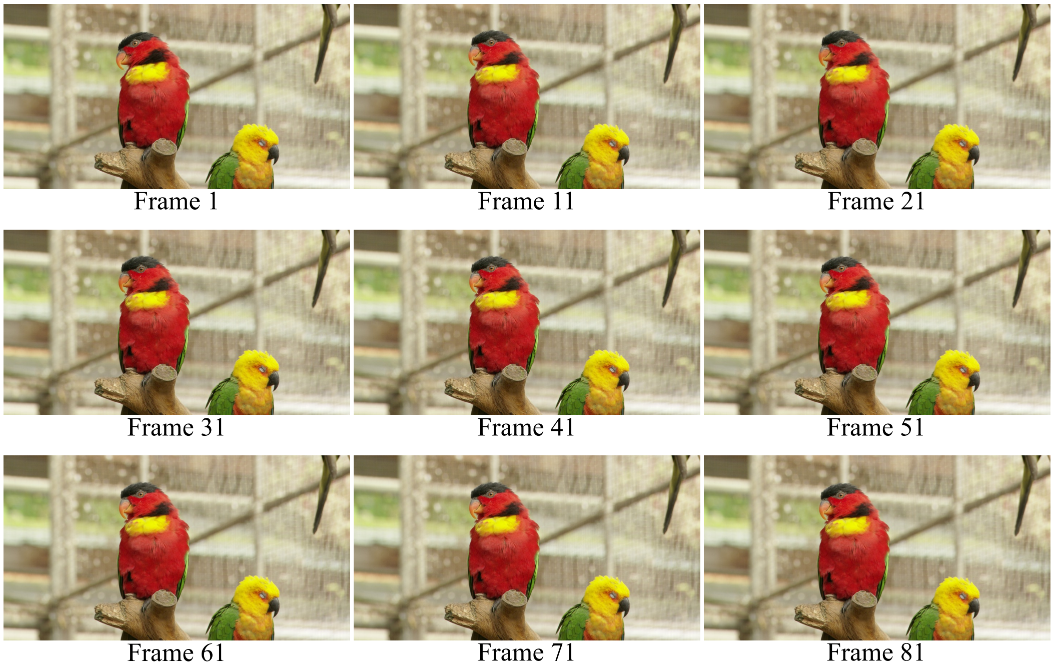
Frame 1
Frame 11
Frame 21
Frame 31
Frame 41
Frame 51
Frame 61
Frame 71
Frame 81

Figure 8: A sequence of frames sampled from video *videoSRC30* of the MCL-JCV dataset, which has high redundancy.

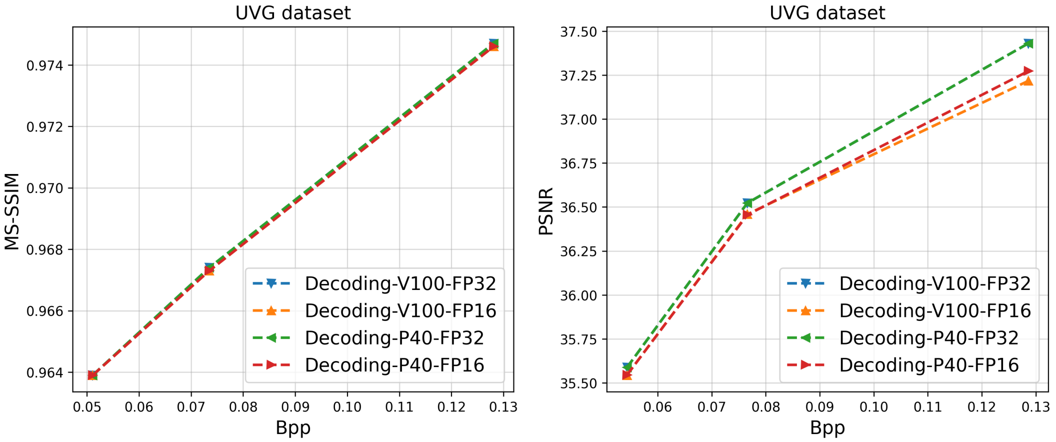

Figure 9: Rate-distortion of our method under different decoding platforms on the UVG dataset.

Table 5: BD-rate calculated on different platforms by *SSIM* and *PSNR* on UVG dataset.

	SSIM-BPP				PSNR-BPP			
Decoder precision	V100 FP32	V100 FP16	P40 FP32	P40 FP16	V100 FP32	V100 FP16	P40 FP32	P40 FP16
UVG	0 %	+0.9 %	0 %	+0.9 %	0 %	+4.2 %	0 %	+3.9 %

C CROSS-PLATFORM ALATION

To verify the cross-platform capability, we encode the video at V100 using our model, which runs with FP32 precision (i.e., single-precision floating point format). On the decoding side, we chose V100 and P40 to decode using FP32 and FP16 (i.e., half-precision floating point format), respectively. In Fig. 9, we plot the rate-distortion of our method under different decoding platforms. For our cross-platform settings, the difference between FP32 and FP16 is very large, and our method is still able to reconstruct all frames correctly, which further demonstrates the cross-platform capability of our approach. Compared to the FP32 decoding results on V100, we calculate the BD-rate of the decoding results under other conditions, which is shown in Table 5.

It is worth noting that the decrease in BD-rate when decoding with FP16 precision, compared to decoding with FP32 precision, is attributed to our model being trained at FP32 precision. This difference in BD-rate is not a cross-platform discrepancy but rather a result of the disparity in computational accuracy between FP16 and FP32.

D WINDOW-BASED CROSS-ATTENTION ABLATION

In Fig. 10, we show the performance of our method when the context model is based on different approaches. It can be seen that our proposed *WCA-based* context model can bring significant improvement compared to the *Resblock-based* context model. It is worth noting that a *light-decoder* model with fewer parameters can decode 1080P video in real-time on V100 and still outperform H.265.

E RATE-DISTORTION CURVES FOR DIFFERENT GOP SIZES.

Larger GOP. Our approach uses a *WCA-based* context model for context fusion, which allows us to employ a larger GOP for compression. We evaluate the UVG dataset at different GOP sizes. The rate-distortion curves are shown in Fig. 11. It has been observed that the model's performance demonstrates a noteworthy enhancement as the GOP size is elevated from 7 to 32. However, there seems to be no substantial alteration in the model's performance as the GOP size is further increased

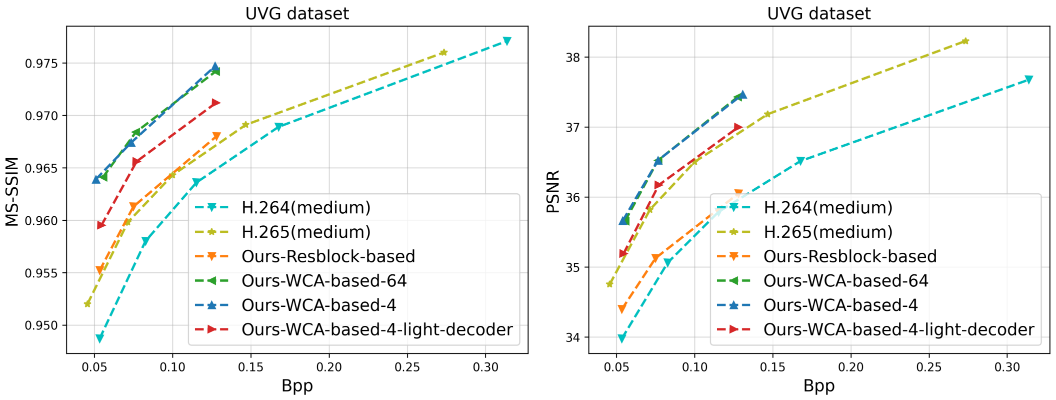

Figure 10: Rate-distortion of different context models on UVG dataset.

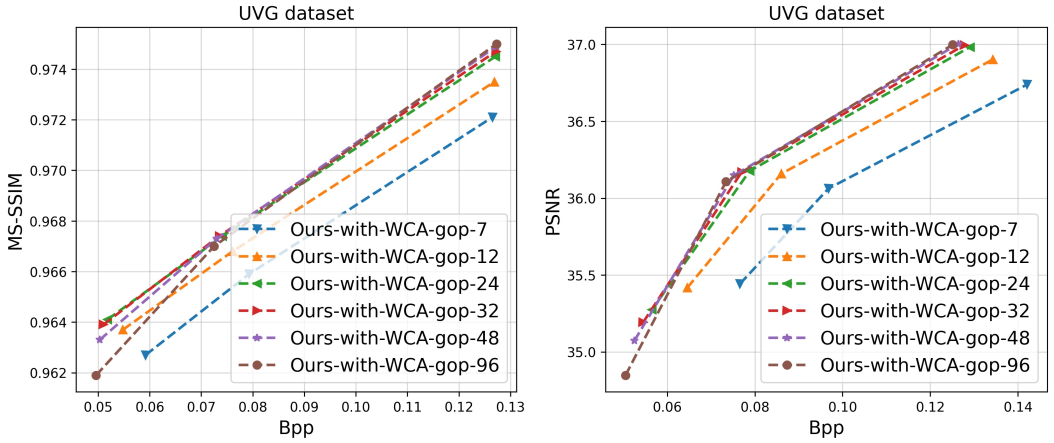

Figure 11: Rate-distortion for different GOP sizes on UVG dataset.

from 32 to 96. Table 6 displays the BD-rate for various GOP sizes, with a GOP size of 32 as the baseline.

F MORE PERFORMANCE COMPARISON.

To further demonstrate the effectiveness of our method, we conduct experiments on additional datasets, including HEVC Class C (480P), D (240P), and E (720P). Beyond PSNR and MS-SSIM, we introduced the perceptual evaluation (Andersson et al., 2020) metric LPIPS (Zhang et al., 2018) as a supplementary experiment. In addition, we compare the performance of different GOP sizes (e.g., 12 and 32), which are commonly used in present methods (Li et al., 2021a; Lu et al., 2019; Li et al., 2022; Tian et al., 2023; Li et al., 2023; Sheng et al., 2022). Furthermore, for H.264 and H.265, we analyze the impact brought by quality control parameters (e.g., *qp* and *crf*). The detailed FFMPEG settings are shown in the Appendix, Section G.

Table 6: BD-rate for different GOP sizes on UVG dataset.

GOP size	7	12	24	32	48	96
SSIM-BPP	+21.9%	+9.0 %	+0.9 %	0 %	-0.2 %	+0.5%
PSNR-BPP	+32.4%	+12.3 %	+1.9 %	0 %	-1.3 %	-1.8 %

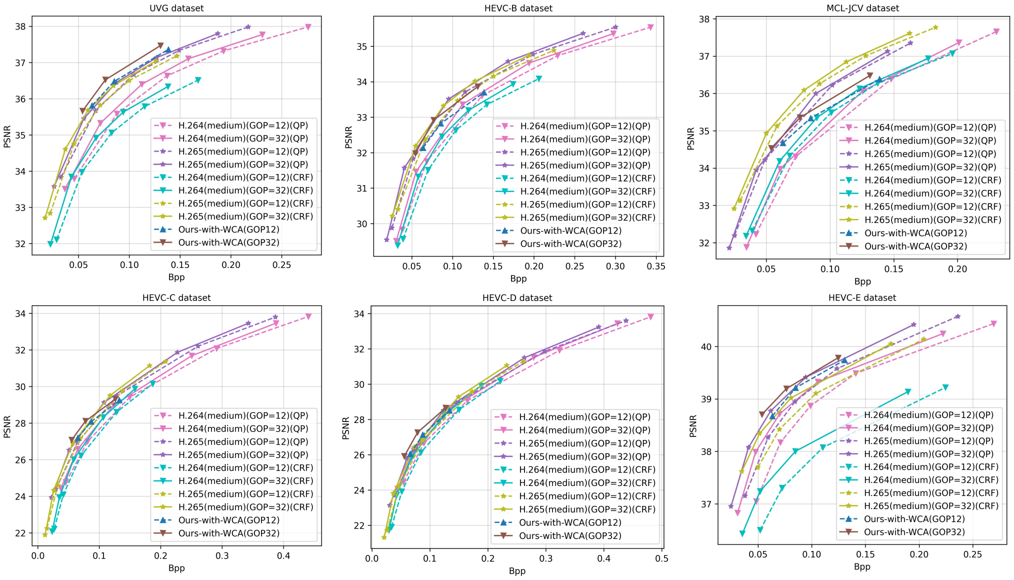

Figure 12: Rate-distortion comparison using PSNR as the metric.

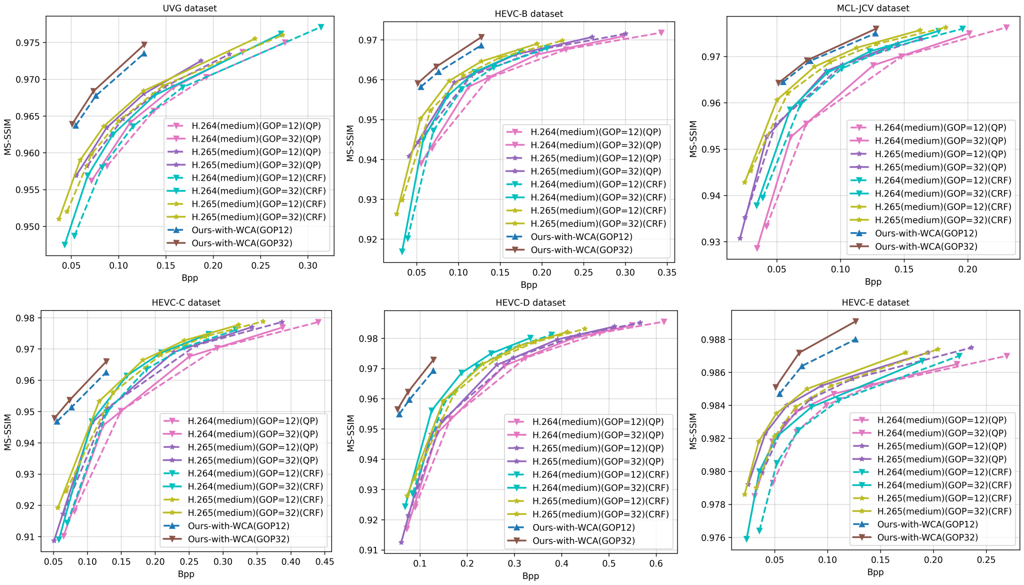

Figure 13: Rate-distortion comparison using MS-SSIM as the metric. Our model is fine-tuned for MS-SSIM.

Table 7: The detailed settings of x264 and x265.

x264-CRF	**x265-CRF**
ffmpeg -pix_fmt yuv420p -s *widthxheight* -i *input_file_name* -c:v libx264 -tune zerolatency -preset medium -x264-params "**crf**=*qp*:keyint=*gop*:verbose=1" *output_video_file_name*	ffmpeg -pix_fmt yuv420p -s *widthxheight* -i *input_file_name* -c:v libx265 -tune zerolatency -preset medium -x265-params "**crf**=*qp*:keyint=*gop*:verbose=1" *output_video_file_name*
x264-QP	**x265-QP**
ffmpeg -pix_fmt yuv420p -s *widthxheight* -i *input_file_name* -c:v libx264 -tune zerolatency -preset medium -x264-params "**qp**=*qp*:keyint=*gop*:verbose=1" *output_video_file_name*	ffmpeg -pix_fmt yuv420p -s *widthxheight* -i *input_file_name* -c:v libx265 -tune zerolatency -preset medium -x265-params "**qp**=*qp*:keyint=*gop*:verbose=1" *output_video_file_name*

G FFMPEG SETTINGS.

In our experiments, the conventional codecs H.264 and H.265 in FFmpeg are used as the anchor with *medium* preset. In current methods(Lu et al., 2019; Li et al., 2021a; 2022), *qp* and *crf* are commonly used for compressing video with quality control, while it is known that *crf* can often achieve the maximum compression efficiency [1]. Therefore, we provide separate configurations for experiments using *qp* and *crf*. The detailed settings of x264 and x265 are shown in Table 7.

[1] https://trac.ffmpeg.org/wiki/Encode/H.264

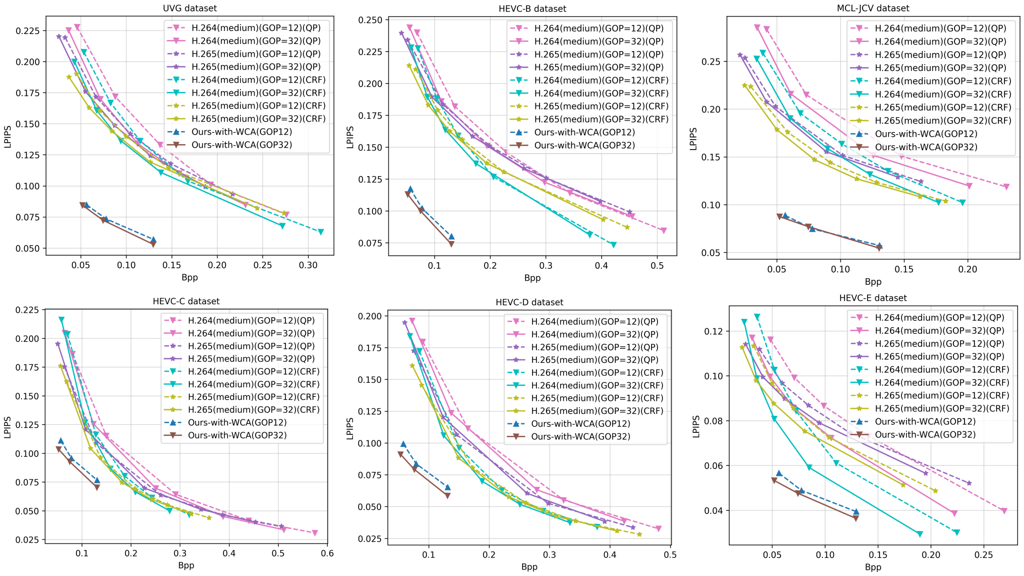

Figure 14: Rate-distortion comparison using LPIPS as the metric. Our model is fine-tuned for LPIPS.