LOW-COMPLEXITY DEEP VIDEO COMPRESSION WITH A DISTRIBUTED CODING ARCHITECTURE

Anonymous authors
Paper under double-blind review

ABSTRACT

Prevalent video compression methods follow a *predictive coding* architecture that relies on a heavy encoder to exploit the statistical redundancy, which makes it challenging to deploy them on resource-constrained devices. Meanwhile, as early as the 1970s, distributed source coding theory, namely, Slepian-Wolf and Wyner-Ziv theorems, has indicated that efficient compression of correlated sources can be achieved by exploiting the source statistics at the decoder only, with the help of effective side information (SI). This has inspired a *distributed coding* architecture that is promising to reduce the encoder complexity. While there have been some attempts to develop practical distributed video coding systems, traditional methods suffer from a substantial performance gap to the predictive coding architecture. Inspired by the recent successes of deep learning in enhancing image and video compression, we propose the first end-to-end distributed deep video compression (Distributed DVC) framework with neural network-based modules that can be optimized to improve the rate-distortion performance. A key ingredient is an effective SI generation module at the decoder, which helps to effectively exploit the inter-frame correlation without computation-intensive encoder-side motion estimation and compensation. Experiments show that Distributed DVC significantly outperforms conventional distributed video coding methods and H.264. Meanwhile, it enjoys $6 \sim 7$ times encoding speedup against DVC (Lu et al., 2019) with only 1.61% increase in the bitrate for 1080P test videos on the UVG dataset.

1 INTRODUCTION

Given the ubiquity and popularity of various video-based applications, deep learning (DL)-based video compression approaches (Lu et al., 2019; Lin et al., 2020; Agustsson et al., 2020; Hu et al., 2021; Li et al., 2021) have attracted increasing attention due to their superior performance over the traditional video codecs represented by H.264 (Wiegand et al., 2003) and H.265 (Sullivan et al., 2012). Most of these novel learning-based methods adopt a predictive coding architecture inherited from popular standard video codecs. As illustrated in Figure 1(a), this paradigm applies a computation-intensive motion compensation prediction loop at the encoder to explicitly reduce temporal redundancy between frames and utilizes several techniques, including transform coding and entropy coding, to lessen the spatial dependency within frames and the statistical correlation of coded symbols. In particular, Table 1 shows that the motion-related operations (*i.e.*, estimation, compensation and compression) contributes about 90% and 65% computation complexity in DVC (Lu et al., 2019) and DCVC (Li et al., 2021), two representative examples of DL-based video codecs. Thus, these methods are characterized by a heavy encoder, while the decoder is relatively simple since it does not need to estimate the motion information from adjacent frames. Such an asymmetry codec architecture is suitable for broadcast-oriented applications such as video-on-demand (Ghose & Kim, 2000), 360^0 video streaming (Fan et al., 2019) and Blu-Ray discs (Miyagawa, 2014), where the videos are encoded once and decoded many times.

Recently there is an upsurge in uplink-based video applications, such as video surveillance (Elharrouss et al., 2021), mobile video chat (Jana et al., 2013), and multi-view image acquisition (Hussain et al., 2021), where the video encoder is deployed on a resource-constrained device. Given the limited onboard computing resources and power supply, such application scenarios demand low-complexity and low-power video encoders, which makes it challenging to deploy existing learning-based video codecs. This calls for a radical change in the video coding architecture. Inspired by

Table 1: Complexity of key components for learning-based video encoders on HD 1080 videos (UVG dataset) running on an Intel Xeon Gold 6230R processor with a base frequency 2.10GHz and a single CPU core.

Latency/FLOPs	Motion			Residual/WZ Compression	Total
	Estimation	Compensation	Compression		
DCVC	13.02s 1253.65G	12.74s 732.83 G	24.15s 1049.95G	29.28s 1557.32G	79.18s 4593.79G
DVC	13.53s 1253.65G	15.90s 783.14G	12.60s 635.04G	3.13s 187.3G	45.16s 2859.12 G
Proposed	0 0	0 0	0 0	6.68s 500.45G	6.68s 500.45G

two notable information theoretical results developed in the 1970s on distributed source coding, i.e., Slepian-Wolf (SW) (Slepian & Wolf, 1973) and Wyner-Ziv (WZ) (Wyner & Ziv, 1976) theorems[1], distributed video coding[2], also known as WZ video coding, has emerged as a promising solution to complement the existing video compression methods for advanced multimedia applications. Specifically, as shown in Figure 1(b), the WZ video codec[3] shifts the computation-intensive motion operations to the decoder and adopts the video interpolation technique to generate the side information (SI) frame $\bar{x}$. Then, an SW codec based on error correcting codes is used to correct bit errors between the low frequency coefficients of the SI transform $\bar{y}$ and that of the original transform $\hat{y}$, while the high frequency components of SI are directly incorporated to the decode frame during reconstruction. In this way, WZ video coding can effectively reduce the computational load of end devices at the expense of a high-complexity decoder while maintaining the compression efficiency.

Nevertheless, it is non-trivial to apply WZ video coding techniques to build a high-efficient and low-complexity practical video compression system. **First**, existing WZ video codecs only achieve the rate-distortion (RD) performance similar with H.264-intra coding (Kodavalla & Mohan, 2011; 2012), but there is a large gap to popular video coding standards, which is predominantly caused by the poor quality of SI at the decoder (Dufaux et al., 2010). It is unclear how to produce and exploit SI at the decoder to implicitly reduce temporal correlations between frames for approaching the performance of predictive coding. **Second**, as shown in Figure 1(b), there is a feedback channel from the decoder to encoder in classical SW codecs (Aaron et al., 2002; Dash et al., 2018; 2019). The decoder has to request additional parity bits repeatedly until the decoding is successful, resulting in a large decoding delay. Meanwhile, it demands an extra frame buffer to store the encoded stream, which consumes high memory at the encoder and is undesirable for mobile devices and cameras (Kodavalla & Mohan, 2010). Although some works explored the feedback channel-free architecture, they typically sacrificed the coding efficiency (Puri & Ramchandran, 2002; Mallick & Mukherjee, 2014; Zhou et al., 2019). To guarantee user experience for video applications, it is of critical importance to develop resource-friendly methods that can efficiently compress videos.

In this paper, we design the first end-to-end distributed deep video coding (Distributed DVC) system to boost the coding efficiency to match the ones with the predictive coding architecture by exploiting the advantages of deep neural networks in nonlinear transform and end-to-end optimization, while alleviating the high encoder complexities of learning-based video codecs by removing the computation-intensive motion operations from the encoder side. Specifically, **to improve the RD performance**, we first propose a lightweight *WZ encoder network* to effectively map frames into their quantized latent representations, which can better capture intra-frame correlations compared with the linear transform. At the decoder, an *SI generation module* is applied to estimate the intermediate motion information and produce the current SI representation from two previous decoded frames and a temporal encoding input, thereby implicitly exploiting the temporal correlation between frames. Assisted by this SI representation, a *WZ decoder network* will reconstruct the video frames based on the quantized representation passed from the encoder. Furthermore, as all

[1]More details about these two theorems are provided in Appendix A.1.

[2]Distributed video coding has been called as DVC in literatures, but as DVC is more often used to refer to deep video compression recently, we call distributed video coding as WZ video coding in this paper.

[3]More details about the WZ video coding are provided in Appendix A.2.

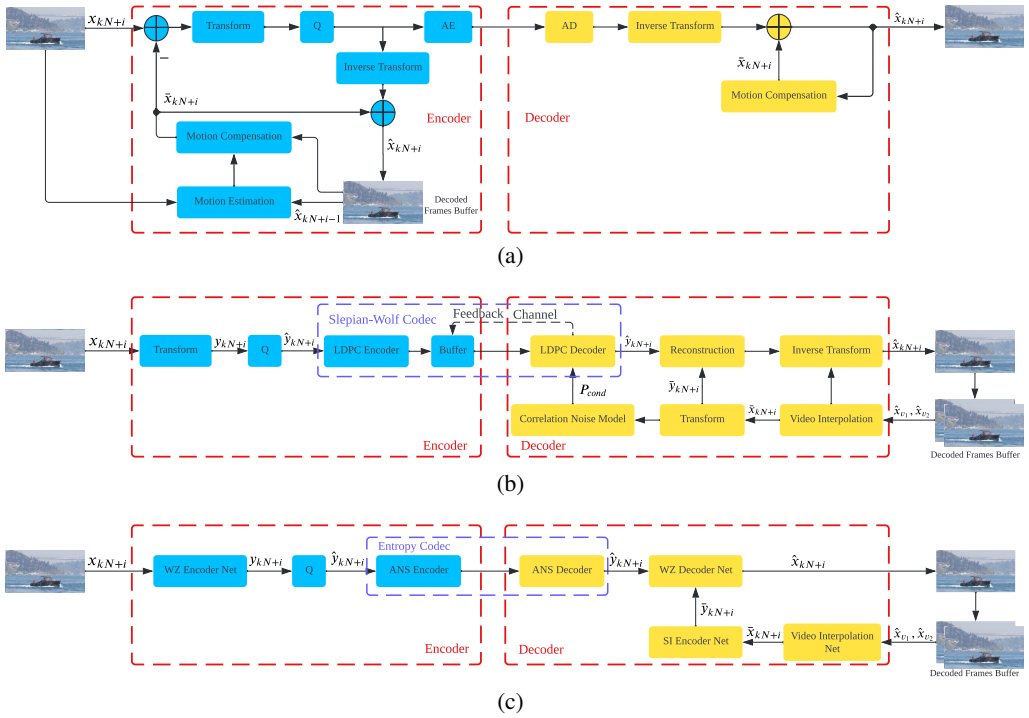


Figure 1: (a) The predictive coding paradigm used by standard video codecs. (b) The traditional WZ video coding architecture. (c) The proposed distributed deep video coding architecture.

the major components are implemented by neural networks, leveraging end-to-end training helps to further improve the RD performance. **To resolve the problems brought by the feedback channel**, a *channel-wise auto-regressive entropy model* (Minnen & Singh, 2020) is used to provide accurate probability modeling for entropy coding of the latent representations, which can effectively reduce the statistical redundancy and avoid the usage of the SW codec. The main contributions of this paper are as follows:

- We develop a deep learning-based distributed video coding architecture to enable low-complexity encoders, which are desirable for uplink-based video applications with resource-constrained devices. While distributed source coding theory has inspired such an architecture decades ago, there is a lack of practical methods that can achieve RD performance close to the predictive coding architecture, and our study fills the gap.
- We design neural network-based modules for key components, including a lightweight WZ encoder for reducing intra-frame redundancy, and a powerful WZ decoder to capture inter-frame correlation, assisted by an effective SI generation module consisting of a video interpolation network and an SI encoder network, as illustrated in Figure 1(c). We also propose a two-step end-to-end training strategy to optimize the RD trade-off.
- Our proposed framework provides up to 10 dB gains over traditional WZ video coding in terms of peak signal-to-noise ratio (PSNR) and it outperforms H.264 on benchmark datasets including UVG and MCL-JCV. Compared with H.265, our method achieves a lower PSNR, but it achieves higher coding gains in multi-scale structural similarity index (MS-SSIM) except for the very low-rate regime. Meanwhile, it reduces about 85% and 90% encoding latency against DVC and DCVC, respectively.

2 Related Works

Learning-based Compression. Recently, learning-based image and video compression methods have achieved significant progresses in terms of RD performance compared with standard codecs,

Figure 2: (a) The hierarchical frame interpolation order. (b) The fast-decoding frame interpolation order.

such as JPEG (Wallace, 1992), BPG (Bellard, 2014), H.264 (Wiegand et al., 2003) and H.265 (Sullivan et al., 2012). Most of the deep image compression methods employ an auto-encoder style network with different types of entropy models for high compression efficiency by utilizing RD optimization techniques (Ballé et al., 2017; 2018b; Minnen et al., 2018; Minnen & Singh, 2020; Cheng et al., 2020; He et al., 2021; Zou et al., 2022). Moreover, other techniques such as generalized divisive normalization (GDN) (Ballé et al., 2016), latent residual prediction and round-based training (Minnen & Singh, 2020) have also been proposed to improve the coding efficiency. We consider these existing works to be important building blocks for our framework.

Existing learning-based video compression methods (Lu et al., 2019; Lin et al., 2020; Agustsson et al., 2020; Hu et al., 2021; Li et al., 2021) mainly follow the predictive coding architecture as shown in Figure 1(a). Thus, they suffer from high complexity in the encoding process due to the computation-intensive motion-related operations. By contrast, our method achieves a better trade-off between the RD performance and encoding complexity, which complements the current learning-based video compression approaches and is suitable for computing resource crunched situations. In addition, different from orthogonal works that apply model quantization to reduce the complexity of learning-based image codecs (Ballé et al., 2018a; Hong et al., 2020; Sun et al., 2021; Yu et al., 2022; Jia et al., 2022), our work aims at reducing the encoder complexity by changing the coding architecture.

Distributed Video Coding. In the past decades, various handcrafted approaches and tools have been proposed to improve the efficiency of WZ video coding, including the application of different transformation (Aaron et al., 2004; Mallick & Mukherjee, 2014), more accurate correlation noise estimation (Brites et al., 2006; Brites & Pereira, 2008) and the refinement of SI (Artigas et al., 2007; Ren et al., 2011). Recently, some works have introduced deep neural networks (DNNs) to boost the performance. Specifically, Bhagath et al. (2016) exploited the learning-based super resolution technique to reconstruct the full resolution frame from the encoded information of half pixels. Dash et al. (2018; 2019) utilized the ensemble of multi-layer perceptron networks and Chebyshev polynomial based functional link artificial neural networks to generate better SI frames. However, in these works, all the modules except the neural network-based one are manually designed, and thus cannot be end-to-end optimized. In contrast, all key components of our proposed framework are implemented with DNNs and jointly optimized to largely improve the RD performance. Moreover, instead of capturing the inter frame correlation by estimating SI at the encoder in Zhou et al. (2022), our proposed method follows the traditional WZ video coding pipeline, which enjoys significant encoding speedup than existing learning-based predictive codecs.

3 Proposed Method

Notations. Let $\mathcal{X} = \{x_1, x_2, ...\}$ denote the original video sequence. WZ video coding applies an adaptive or fixed frame separator to split the video sequence into Key frames and WZ frames. For simplicity, we assume a fixed size of group of pictures (GOP) as N. In this case, x_{kN+1} represents a key frame of the video sequence, and the other frames x_{kN+i} are WZ frames, where $k = \{0, 1, 2, ...\}$ and $i = \{2, 3, ..., N\}$. $\hat{x}_{kN+i}$ denotes the reconstructed WZ frame. In order to reduce the temporal redundancy, an SI frame $\bar{x}_{kN+i}$ is generated using two previous decoded frames $\hat{x}_{v_1}$ and $\hat{x}_{v_2}$ at the decoder, where v_1 and v_2 denote the index of the reference frames. Transform coding can be used to improve the compression efficiency. In such case, the original frame x_{kN+i} and SI frame $\bar{x}_{kN+i}$ are transformed to y_{kN+i} and $\bar{y}_{kN+i}$, respectively. $\hat{y}_{kN+i}$ is the quantized version of y_{kN+i}.

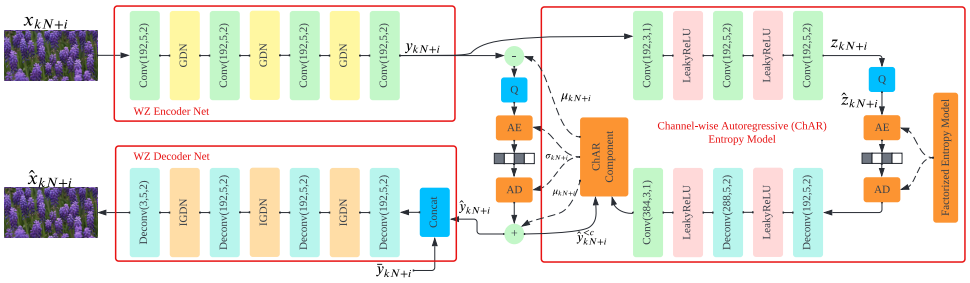


Figure 3: Proposed WZ encoder-decoder network. Convolution parameters are formatted as (the number of filters, kernel size, stride). AE and AD denote ANS encoder and decoder, respectively.

3.1 Overview of the Proposed Method

Figure 1(c) presents a high-level overview of the proposed end-to-end Distributed DVC framework, and the key differences from the classic WZ video codec are explained as follows:

Step 1. Transformation and quantization. We replace the linear transformation by a non-linear encoder, i.e., a WZ encoder network, and the input WZ frame x_{kN+i} is non-linearly mapped to the representation y_{kN+i}. Then y_{kN+i} is quantized to $\hat{y}_{kN+i}$. To enable end-to-end training, we adopt the quantization method in Minnen & Singh (2020). Details are presented in Section 3.2.

Step 2. Entropy encoding. Instead of using an SW codec to compress the quantized WZ frame, we adopt an asymmetrical numeral systems (ANS) encoder (Duda, 2013) to encode the quantized WZ representation $\hat{y}_{kN+i}$ into bits at the inference stage. At the training stage, to estimate the number of bits cost in our proposed approach, we use an entropy model to estimate the probability distribution of each symbol in $\hat{y}_{kN+i}$. Details are given in Section 3.2.

Step 3. Side information generation. An optical flow-based video interpolation network in Huang et al. (2020) is adopted to estimate the current SI frame $\bar{x}_{kN+i}$ based on two previous decoded frames $\hat{x}_{v_1}$ and $\hat{x}_{v_2}$ with the hierarchical/fast-decoding interpolation order shown in Figure 2. Furthermore, an SI encoder is designed to produce the SI representation $\bar{y}_{kN+i}$. More information is provided in Section 3.3.

Step 4. Entropy decoding. The decoder receives the bit stream from the encoder and performs ANS decoding to reconstruct the quantized WZ representation $\hat{y}_{kN+i}$. Compared with SW decoding, the entropy decoding is more efficient, and it helps to get rid of the feedback channel of the traditional architecture.

Step 5. Inverse transformation. We concatenate the decoded WZ representation $\hat{y}_{kN+i}$ and SI representation $\bar{y}_{kN+i}$, and feed them into the decoder network to reconstruct the WZ frame $\hat{x}_{kN+i}$, rather than using a predefined quantization table to perform the reconstruction process. Details are provided in Section 3.2.

3.2 WZ Encoder and Decoder Networks

To facilitate the compression of the WZ frame x_{kN+i}, while assisting the generation of the SI at the decoder to reconstruct the WZ frame $\hat{x}_{kN+i}$, we design a CNN-based WZ encoder-decoder network, taking inspiration from deep image compression (Ballé et al., 2018b; Minnen et al., 2018). As illustrated in Figure 3, given an input frame x_{kN+i}, the WZ encoder generates the representation y_{kN+i} that is quantized to $\hat{y}_{kN+i}$. The WZ decoder receives the quantized representation from the encoder and reconstructs the WZ information $\hat{x}_{kN+i}$ with the aid of the SI representation $\bar{y}_{kN+i}$ from the SI generation module. We employ the channel-wise auto-regressive (ChAR) entropy model (Minnen & Singh, 2020) to estimate the probability distribution of y_{kN+i}. Specifically, a hyper prior entropy model (Ballé et al., 2018b) generates the hyper representation $\hat{z}_{kN+i}$ to capture the spatial redundancies among the elements of $\hat{y}_{kN+i}$. Besides, a ChAR component is utilized to shrink the correlations among the channels of $\hat{y}_{kN+i}$ by exploiting the casual context. These two ingredients generate the mean and scale parameters for a conditional Gaussian entropy model. In addition, since the quantization operation is not differentiable, it renders gradient descent ineffective and hinders the end-to-end network training. In order to allow optimization via stochastic gradient descent, we apply

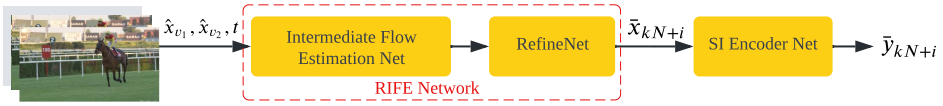


Figure 4: Side information generation network.

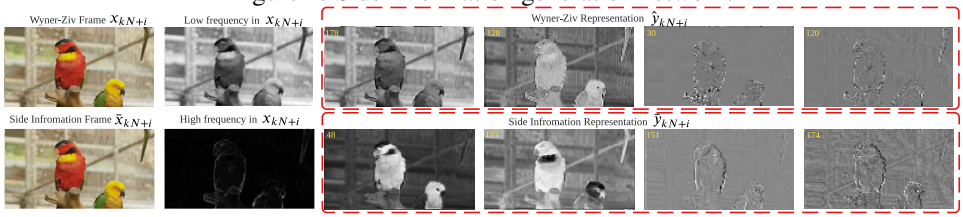


Figure 5: Visualization results from the *videoSRC30* sequence in MCL-JCV dataset. The Fourier transform is used to decompose the low frequency and high frequency content in x_{kN+i}. We separately select two low-frequency and high-frequency channels for visualization.

the mixed quantization method proposed in Minnen & Singh (2020). In particular, the quantizer is replaced with an additive uniform noise for learning entropy model, and a rounded tensor with straight-through gradient replaces the noise one to flow to the WZ decoder. Moreover, two variants (lite/pro versions) of our WZ auto-encoder style networks are proposed for speed/efficiency priority, with more details provided in Appendix B.

3.3 Side Information Generation Network

The SI generation network is composed of two parts, i.e., an optical flow-based video interpolation network and an SI encoder. The overall architecture of the proposed network is shown in Figure 4. We adopt the RIFE network (Huang et al., 2020) for video interpolation that allows arbitrary time-step frame interpolation with two previous decoded frames $\hat{x}_{v_1}$ and $\hat{x}_{v_2}$. Thus, the proposed framework supports hierarchical and fast-decoding interpolation orders, as shown in Figure 2, which can satisfy the needs of different applications. Specifically, the former achieves better coding efficiency with higher SI frame quality, and the latter is parallelization-friendly so it can pursue the fast decoding of WZ frames without relying on other frames except key frames. After obtaining the current SI frame $\bar{x}_{kN+i}$, we utilize the SI encoder with the same network as the WZ encoder described in Section 3.2 to produce the SI representation $\bar{y}_{kN+i}$. Thus, the SI encoder also has two different variants, i.e., lite, and pro versions. More details of the network in Figure 4 are provided in Appendix B.

To better understand the difference between SI and WZ representations, we provide the visualization of feature maps in Appendix C.5, where some channel-wise activations are shown in Figure 5. From the visualization results, we observe that when compared with the SI representation $\bar{y}_{kN+i}$, there are more (fewer) channels in $\hat{y}_{kN+i}$, e.g., the 128-th (30-th) and 178-th (170-th) channels, to represent low-frequency (high-frequency) information. Besides, only a few channels in $\bar{y}_{kN+i}$, e.g., the 48-th and 123-th channels, emphasize the low-frequency content, while most of channels of the SI representation, like the 151-th and 174-th channels, seem to pay more attention to the high-frequency details of the birds' edges and silhouettes in contrast with high frequency in x_{kN+i}. This matches the reconstruction operation of the traditional WZ video coding where the high frequency DCT coefficients of the SI frame are directly used as that of the reconstructed WZ frame.

3.4 Training Strategy

Loss Function. Our design objective is to minimize the number of encoded bits and reduce the distortion between the original WZ frame x_{kN+i} and the reconstructed WZ frame $\hat{x}_{kN+i}$. Thus, we use the following loss function consisting of two metrics for training:

$$L = \lambda D + R = \lambda d(x_{kN+i}, \hat{x}_{kN+i}) + R(\hat{y}_{kN+i}) + R(\hat{z}_{kN+i}) \tag{1}$$

where $d(x_{kN+i}, \hat{x}_{kN+i})$ is the distortion between x_{kN+i} and $\hat{x}_{kN+i}$, which can be mean squared error (MSE) or MS-SSIM (Wang et al., 2003) for different tasks. $R(\hat{y}_{kN+i})$ and $R(\hat{z}_{kN+i})$ denote the bit rates used for encoding the quantized WZ representation $\hat{y}_{kN+i}$ and the corresponding hyper

representation $\hat{z}_{kN+i}$, respectively. λ is a Lagrange multiplier that controls the trade-off between the bit rate cost R and distortion D.

Two-step Training. During training, we adopt a two-step procedure for MSE optimized models. Firstly, we train the WZ encoder-decoder and SI encoder networks, while the RIFE network adopts the pretrained model in Huang et al. (2020) with fixed parameters. After that, we jointly fine-tune the whole model including the RIFE network. For MS-SSIM optimized models, we fine-tune all of the modules based on MSE optimized networks.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Training Data. We use the training part of the Vimeo-90k dataset (Xue et al., 2019) to train the proposed video compression framework, and randomly crop the videos into 256×256 patches. During training, we set the mini-batch size as 16 (i.e., 16 video clips).

Test Settings. We evaluate the compression performance of our proposed method on two video test datasets with diversified content, including the UVG dataset (Mercat et al., 2020) and MCL-JCV dataset (Wang et al., 2016). The GOP size is set as 8 by default. We compare our method using the hierarchical interpolation order with the traditional codecs, including WZ video coding (Kodavalla & Mohan, 2012), H.264 (Wiegand et al., 2003) and H.265 (Sullivan et al., 2012), as well as learning-based methods, such as DVC (Lu et al., 2019), DVC-Lite (Lu et al., 2020) and DCVC (Li et al., 2021). For H.264 and H.265, we report the only-P mode and hierarchical-B mode. In addition, two I-frame codecs Minnen & Singh (2020) and Ballé et al. (2018b) are evaluated to demonstrate the effectiveness of the SI generation network. For details about H.264/H.265 settings, please refer to Appendix C.1.

Evaluation Metrics. Both PSNR and MS-SSIM are used to measure the distortion of the reconstructed frames, and bits per pixel (bpp) is used to measure the number of bits for encoding the WZ representation. Additionally, the Bjøntegaard Delta bitrate (BDBR) (Bjøntegaard, 2001) is computed to denote the average bitrate savings at the same reconstruction quality.

Implementation Details. In our experiments, we use the pretrained model *mbt-2018* (Minnen et al., 2018) provided by CompressAI (Bégaint et al., 2020) for key frame compression. For WZ frame compression, we train five models with different λ values (MSE: 0.0018, 0.0035, 0.0067, 0.0130, 0.0250; MS-SSIM: 2.40, 4.58, 8.73, 16.64, 31.73) for multiple coding rates. In addition, Adam (Kingma & Ba, 2014) is used with an initial learning rate as 0.001 that is reduced by a factor of 2 when the evaluation loss reaches a plateau. We use a patience of 10 epochs and 5 epochs for the first and second stage, respectively. As for the total number of epochs, we set it as 100 and 50 for the two stages respectively. The whole system is implemented by PyTorch and trained on an NVIDIA RTX A5000 GPU.

4.2 EXPERIMENTAL RESULTS

RD Performance. Figure 6 presents the compression performance of different methods. The proposed methods outperform the traditional WZ video coding by a large margin with $9.43 \sim 10.29$ dB and $6.75 \sim 7.24$dB gains in terms of PSNR and MS-SSIM, which implies that end-to-end optimization design can effectively improve the performance of the distributed coding architecture. When compared with the I-frame codec Minnen2020 (Ballé2018), the proposed (lite) method improves about 40% and 20% coding efficiency on the UVG dataset and MCL-JCV datasets in PSNR, which indicates that using the SI at decoder can implicitly reduce the temporal redundancy between frames to some extent. Compared with H.264(P) and H.264(B), the proposed method saves 24.30% (22.63%) and 13.13% (13.03%) bits in terms of PSNR on the UVG (MCL-JCV) dataset, respectively. Compaerd with H.265, our methods achieves a lower PSNR, but it achieves better performance under MS-SSIM except for very low-rate regime. Moreover, the proposed method attains better and similar coding gains than DVC-Lite and DVC on the UVG dataset, respectively. The proposed methods underperform the current SOTA video codec DCVC, which is partly because the temporal correlation is not exploited at the encoder. More experimental results for per video are given in Appendix C.4.

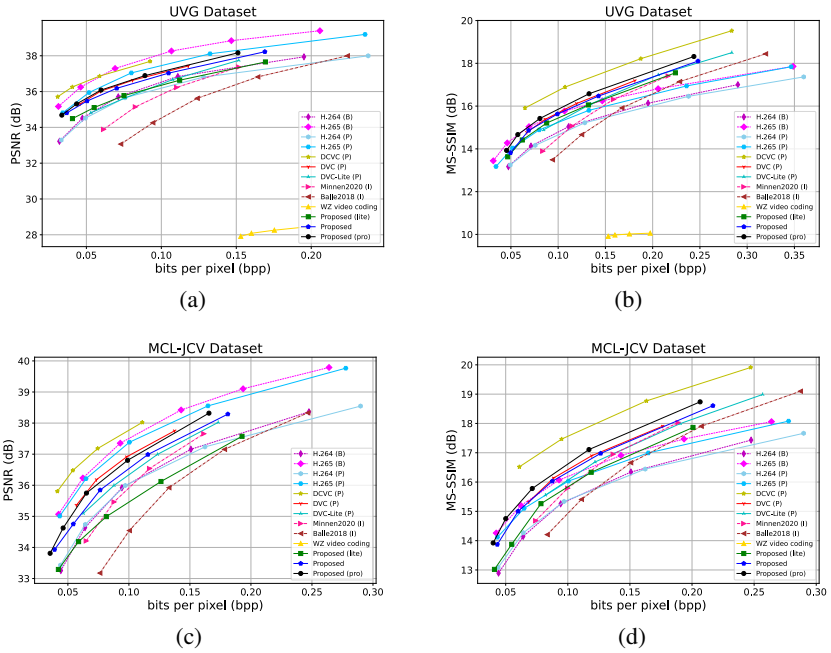


Figure 6: Rate-Distortion performance in terms of PSNR and MS-SSIM. Due to the inefficiency of the serial implementation of the WZ video codec, we only report its performance on the UVG dataset.

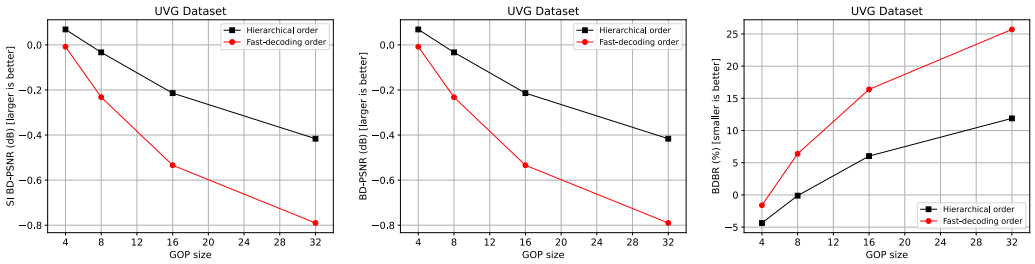
Table 2: Encoding complexity of learning-based codecs. N_c denotes the number of CPU cores used to encode the videos of UVG dataset. H.264 (P) with encoding time as 0.043s under N_c=1 is set as the anchor to measure BDBR.

Method	FLOPs	Latency					UVG (BDBR)	
		N_c=1	N_c=2	N_c=4	N_c=8	GPU	PSNR	MS-SSIM
DCVC	4593.79G	79.18s	42.70s	26.71s	18.61s	–	-55.09%	-63.18%
DVC	2859.12G	45.16s	24.40s	15.51s	9.38s	0.442s	-25.91%	-38.25%
DVC-Lite	1062.14G	20.69s	12.28s	7.82s	4.86s	0.306s	-4.30%	-22.46%
Proposed (pro)	1654.7G	19.48s	10.04s	5.70s	3.64s	0.344s	-32.35%	-41.90%
Proposed	500.45G	6.68s	3.98s	2.29s	1.34s	0.180s	-24.30%	-34.72%
Proposed (lite)	176.75G	3.06s	1.90s	1.09s	0.65s	0.127s	1.01%	-27.48%

Encoder Complexity. We compare the encoding complexity of six video codecs on the 1080P test videos of the UVG dataset on both CPU and GPU devices. The results are shown in Table 2, including the number of FLOPs, the encoding time under different computing capability constraints, and the BDBR performance. On CPU platforms with different computing powers (i.e., N_c=1,2,4,8), the proposed method achieves $3 \sim 3.6$ times encoding speedup against DVC-Lite while saving 20% (12.2%) bits measured by PSNR (MS-SSIM). When compared with DVC, our method achieves $6 \sim 7$ times speedup, only increasing 1.61% bits in PSNR. When the encoder is implemented on a powerful GPU, the proposed method saves 60% inference time in the encoding period. Furthermore, the pro variant with the better coding efficiency is $2\times$ faster than DVC. Although there is a performance gap between the proposed method and DCVC, our methods can reduce about 90% of the encoding latency. This set of results show our methods based on the distributed coding paradigm have the potential to reach the encoding time of standard video codecs such as H.264. Details on decoding complexity and analysis of each component complexity are in Appendix C.2 and Appendix C.3, respectively.

4.3 Ablation Study

Frame Interpolation Order. As mentioned in Section 3.3, our method supports two different interpolation orders. To investigate their effect, we report the SI and reconstructed frame gains



(a) (b) (c)

Figure 7: Side information frame gain, reconstructed frame gain and bitrate savings at different GOP settings. A GOP size of 2 is set as the benchmark.

Table 3: Ablation study on side information, where the proposed method is selected as the anchor.

RIFE network	SI Encoder	Joint Training	Bitrate increase	PSNR decrease
✓	✓	✓	0%	0dB
✓	✓		9.71%	0.21dB
	✓	✓	24.67%	0.75dB
✓		✓	48.98%	1.24dB

as well as average bitrate savings under different GOP sizes on the UVG dataset. From Figure 7(a), we observe that a large GOP causes the RIFE network to generate low-quality SI frames. The hierarchical interpolation order alleviates this phenomena by exploiting the information from the near decoded frames to interpolate the small time-step SI frame to effectively reduce the warping error. Therefore, the coding gain gap between the hierarchical order and the fast-decoding one becomes larger as the GOP size increases, as shown in Figure 7(b) and 7(c).

Side Information. In our framework, we propose using the RIFE network and SI encoder with the joint training strategy to generate the SI representation. To verify their effectiveness, we conduct the ablation experiments on the UVG dataset. As shown in Table 3, fixing the RIFE network leads to 9.71% increase in bitrate and 0.21dB decrease in PSNR, which can be explained by the fact that the pretrained RIFE network aims at only for estimating the intermediate frame more accurately, but not for optimizing the rate-distortion performance. In addition, if we do not generate the intermediate frame and concatenate two decoded frames following the hierarchical order as the SI, it requires 24.67% more bitrate and drops PSNR by 0.614dB, which verifies the benefit of video interpolation. Moreover, the SI exploited in the pixel space brings up to 48.98% bitrate consumption and 1.24dB gain decrease, supporting the necessity of processing SI in the feature space.

5 Conclusions and Discussions

In this paper, we proposed an end-to-end distributed deep video compression framework to enhance the RD performance of the distributed video coding architecture. Our proposal combines the merits of traditional distributed video coding in the low encoding complexity and learning-based compression in the powerful non-linear representation ability. Experimental results demonstrate the competence of the proposed framework in achieving a better coding efficiency than traditional distributed video coding methods and H.264. In the meantime, compared with DVC, our proposed method enjoys a much lower encoder complexity with a slight increase in the bitrate. Overall, the proposed Distributed DVC framework is promising in enabling deep video compression systems with low-complexity encoders. While there is still a performance gap to deep video compression with the predictive coding architecture, we believe it can be narrowed by leveraging the latest advancements in deep learning to further improve the coding efficiency. For example, scale flow (Agustsson et al., 2020) and deformable compensation (Hu et al., 2021) can be applied to improve the generation of SI. Moreover, Distributed DVC enjoys great potentials in application scenarios with multiple video sources captured by different camera sensors, where only the decoder can exploit the statistical redundancy. Thus, it deserves more research efforts.

REFERENCES

Anne Aaron, Rui Zhang, and Bernd Girod. Wyner-ziv coding of motion video. In *Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems and Computers, 2002.*, volume 1, pp. 240–244. IEEE, 2002.

Anne Aaron, Shantanu D Rane, Eric Setton, and Bernd Girod. Transform-domain wyner-ziv codec for video. In *Visual Communications and Image Processing 2004*, volume 5308, pp. 520–528. International Society for Optics and Photonics, 2004.

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.

Xavi Artigas, Joao Ascenso, Marco Dalai, Sven Klomp, Denis Kubasov, and Mourad Ouaret. The discover codec: architecture, techniques and evaluation. In *picture coding symposium (PCS'07)*, number CONF, 2007.

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. Density modeling of images using a generalized normalization transformation. In *International Conference on Learning Representations*, 2016.

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. In *International Conference on Learning Representations*, 2017.

Johannes Ballé, Nick Johnston, and David Minnen. Integer networks for data compression with latent-variable models. In *International Conference on Learning Representations*, 2018a.

Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *International Conference on Learning Representations*, 2018b.

Jean Bégaint, Fabien Racapé, Simon Feltman, and Akshay Pushparaja. Compressai: a pytorch library and evaluation platform for end-to-end compression research. *arXiv preprint arXiv:2011.03029*, 2020.

Fabrice Bellard. Bpg image format. `https://bellard.org/bpg/`, 2014.

P Raj Bhagath, Jayanta Mukherjee, and Sudipta Mukopadhayay. Low complexity encoder for feedback-channel-free distributed video coding using deep convolutional neural networks at the decoder. In *Proceedings of the Tenth Indian Conference on Computer Vision, Graphics and Image Processing*, pp. 1–7, 2016.

G. Bjøntegaard. Calculation of average psnr differences between rd-curves. *ITU-T VCEG-M33*, 2001.

Catarina Brites and Fernando Pereira. Correlation noise modeling for efficient pixel and transform domain wyner–ziv video coding. *IEEE Transactions on Circuits and systems for Video Technology*, 18(9):1177–1190, 2008.

Catarina Brites, Joao Ascenso, and Fernando Pereira. Studying temporal correlation noise modeling for pixel based wyner-ziv video coding. In *2006 International Conference on Image Processing*, pp. 273–276. IEEE, 2006.

Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7939–7948, 2020.

Bodhisattva Dash, Suvendu Rup, Anjali Mohapatra, Banshidhar Majhi, and MNS Swamy. Decoder driven side information generation using ensemble of mlp networks for distributed video coding. *Multimedia Tools and Applications*, 77(12):15221–15250, 2018.

Bodhisattva Dash, Suvendu Rup, Anjali Mohapatra, Banshidhar Majhi, and MNS Swamy. Decoder side wyner–ziv frame estimation using chebyshev polynomial-based flann technique for distributed video coding. *Multidimensional Systems and Signal Processing*, 30(3):1031–1061, 2019.

Jarek Duda. Asymmetric numeral systems: entropy coding combining speed of huffman coding with compression rate of arithmetic coding. *arXiv preprint arXiv:1311.2540*, 2013.

Frederic Dufaux, Wen Gao, Stefano Tubaro, and Anthony Vetro. Distributed video coding: trends and perspectives. *EURASIP Journal on Image and Video Processing*, 2009:1–13, 2010.

Omar Elharrouss, Noor Almaadeed, and Somaya Al-Maadeed. A review of video surveillance systems. *Journal of Visual Communication and Image Representation*, 77:103116, 2021.

Ching-Ling Fan, Wen-Chih Lo, Yu-Tung Pai, and Cheng-Hsin Hsu. A survey on 360 video streaming: Acquisition, transmission, and display. *ACM Computing Surveys (CSUR)*, 52(4):1–36, 2019.

Debasish Ghose and Hyoung Joong Kim. Scheduling video streams in video-on-demand systems: A survey. *Multimedia Tools and Applications*, 11(2):167–195, 2000.

B. Girod, A.M. Aaron, S. Rane, and D. Rebollo-Monedero. Distributed video coding. *Proceedings of the IEEE*, 93(1):71–83, 2005. doi: 10.1109/JPROC.2004.839619.

Dailan He, Yaoyan Zheng, Baocheng Sun, Yan Wang, and Hongwei Qin. Checkerboard context model for efficient learned image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14771–14780, 2021.

Weixin Hong, Tong Chen, Ming Lu, Shiliang Pu, and Zhan Ma. Efficient neural image decoding via fixed-point inference. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(9): 3618–3630, 2020.

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*, pp. 1502–1511, 2021.

Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. Rife: Real-time intermediate flow estimation for video frame interpolation. *arXiv preprint arXiv:2011.06294*, 2020.

Tanveer Hussain, Khan Muhammad, Weiping Ding, Jaime Lloret, Sung Wook Baik, and Victor Hugo C de Albuquerque. A comprehensive survey of multi-view video summarization. *Pattern Recognition*, 109:107567, 2021.

Shraboni Jana, Amit Pande, An Chan, and Prasant Mohapatra. Mobile video chat: issues and challenges. *IEEE Communications Magazine*, 51(6):144–151, 2013.

Chuanmin Jia, Xinyu Hang, Shanshe Wang, Yaqiang Wu, Siwei Ma, and Wen Gao. Fpx-nic: An fpga-accelerated 4k ultra-high-definition neural video coding system. *IEEE Transactions on Circuits and Systems for Video Technology*, 2022.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Vijay Kumar Kodavalla and PG Krishna Mohan. Distributed video coding (dvc): Challenges in implementation and practical usage. *IP-SOC 2010*, 2010.

Vijay Kumar Kodavalla and PG Krishna Mohan. Chroma components coding in feedback-free distributed video coding. In *2011 IEEE 5th International Conference on Internet Multimedia Systems Architecture and Application*, pp. 1–6. IEEE, 2011.

Vijay Kumar Kodavalla and PG Krishna Mohan. Chroma components coding method in distributed video coding. In *2012 International Conference on Devices, Circuits and Systems (ICDCS)*, pp. 413–417. IEEE, 2012.

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

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.

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.

Kallol Mallick and Jayanta Mukherjee. Distributed video coding using local rank transform. *Proceedings of the 2014 Indian Conference on Computer Vision Graphics and Image Processing*, 2014.

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.

David Minnen and Saurabh Singh. Channel-wise autoregressive entropy models for learned image compression. In *2020 IEEE International Conference on Image Processing (ICIP)*, pp. 3339–3343. IEEE, 2020.

David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. *Advances in neural information processing systems*, 31, 2018.

Naoyasu Miyagawa. Overview of blu-ray disc™ recordable/rewritable media technology. *Frontiers of Optoelectronics*, 7(4):409–424, 2014.

Rohit Puri and Kannan Ramchandran. Prism: A new robust video coding architecture based on distributed compression principles. In *Proceedings of the annual allerton conference on communication control and computing*, volume 40, pp. 586–595. Citeseer, 2002.

Peiming Ren, Ping Shi, Chao Luo, and Qingqing Liu. A new scheme for side information generation in dvc by using optical flow algorithm. In *2011 International Conference on Multimedia Technology*, pp. 2852–2856. IEEE, 2011.

D. Slepian and J. Wolf. Noiseless coding of correlated information sources. *IEEE Transactions on Information Theory*, 19(4):471–480, 1973. doi: 10.1109/TIT.1973.1055037.

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*, 22(12):1649–1668, 2012.

Heming Sun, Lu Yu, and Jiro Katto. End-to-end learned image compression with quantized weights and activations. *arXiv preprint arXiv:2111.09348*, 2021.

Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

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, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pp. 1398–1402. Ieee, 2003.

Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7): 560–576, 2003.

A. Wyner and J. Ziv. The rate-distortion function for source coding with side information at the decoder. *IEEE Transactions on Information Theory*, 22(1):1–10, 1976. doi: 10.1109/TIT.1976.1055508.

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

Hongjiu Yu, Qiancheng Sun, Jin Hu, Xingyuan Xue, Jixiang Luo, Dailan He, Yilong Li, Pengbo Wang, Yuanyuan Wang, Yaxu Dai, et al. Evaluating the practicality of learned image compression. *arXiv preprint arXiv:2207.14524*, 2022.

Junwei Zhou, Yincheng Fu, Yanchao Yang, and Anthony TS Ho. Distributed video coding using interval overlapped arithmetic coding. *Signal Processing: Image Communication*, 76:118–124, 2019.

Junwei Zhou, Ting Lv, and XiangBo Yi. End-to-end distributed video coding. In *2022 Data Compression Conference (DCC)*, pp. 496–496. IEEE, 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*, 2022.

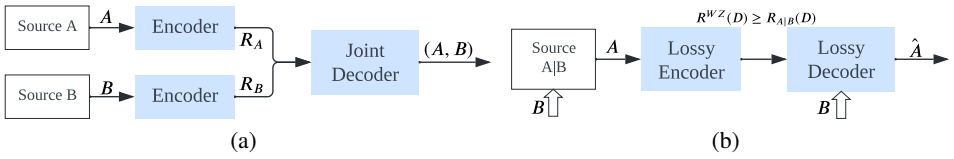


Figure 8: (a): Distributed compression of two correlated sources A and B. The decoder jointly decompress A and B to utilize their mutual dependence. (b): Lossy compression of a source A using statistically related side information B.

A Distributed Coding

A.1 Foundation

As shown in Figure 8(a), distributed coding refers to separate encoding and joint decoding for two (or more) statistically correlated but physically separated sources. The joint decoding procedure aims at exploiting statistical correlations across different sources to achieve efficient compression. Slepian and Wolf studied the distributed lossless coding problem in the basic two-source case in 1973. Wyner and Ziv then extended the Slepian and Wolf (SW) theorem to the lossy case, namely the Wyner–Ziv (WZ) theorem, which presents the achievable lower bound for the bit rate at given distortion, as shown in Figure 8(b). This architecture has been extended to the video coding area, called as distributed video coding (Girod et al., 2005), which aims at independent encoding of each frame and joint decoding with side information generated from previously decoded frames. The formal statements of SW theorem and WZ theorem are as follows.

Theorem 1 (Slepian-Wolf) *Consider two statistically dependent i.i.d. sources A and B, the achievable rate region of compressing A and B without any distortion, is provided by:*

$$R_A \geq H(A|B), R_B \geq H(B|A), R_A + R_B \geq H(A) + H(B),$$

where R_A and R_B represent the rates for transmitting A and B, respectively.

Theorem 2 (Wyner-Ziv) *Assume sources A and B are statisctially correlated. Given a certain distortion level D, the minimum rate $R_{WZ}(D)$ for encoding A with side information B at the decoder is larger than or equal to $R_{A|B}(D)$, where B is available at both the encoder and the decoder. Denoting the output of decoder as $\hat{A}$, we have the following equivalence:*

$$R_{WZ}(D) \geq R_{A|B}(D) = \min_{E[d(A,\hat{A})] \leq D} I(A; \hat{A}|B),$$

where $R_{WZ}(D) = R_{A|B}(D)$ when A and B are jointly Gaussian, and the distortion $d(A, \hat{A})$ between A and $\hat{A}$ is measured by the mean-squared error.

The SW theorem proclaims that lossless compression of two correlated data sources with separate encoders and a joint decoder can asymptotically achieve the same compression rate as the optimal compression with a joint encoder and decoder. The WZ theorem extends this idea to lossy compression and demonstrates that there is no RD performance loss without the side information at the encoder. The SW and WZ theorems imply that it is possible to compress two statistically dependent signals in a distributed way (separate encoding and joint decoding) while reaching the RD performance of predictive coding methods (joint encoding and decoding). For more details on distributed coding and its applications in video coding, please refer to Girod et al. (2005) and Dufaux et al. (2010).

A.2 Brief Introduction of Classic WZ Video Coding

Figure 1(b) illustrates the classic architecture of WZ video coding (Kodavalla & Mohan, 2012). The encoding and decoding procedure of the WZ video compression is briefly summarized as follows,

Step 1. Transformation and quantization. The input WZ frame x_{kN+i} is transformed to y_{kN+i} by applying a block-based transform, *e.g.*, discrete cosine transform (DCT). Then y_{kN+i} is uniformly

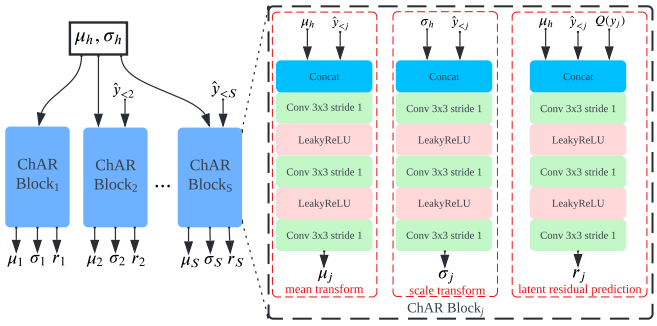


Figure 9: Network structure of ChAR component, where μ_h and σ_h represent the estimated mean and variance from the hyperprior entropy model. The S ChAR blocks are performed sequentially since $\hat{y}_j$ is decoded after μ_j and σ_j is obtained. We set S as 8 in the proposed model.

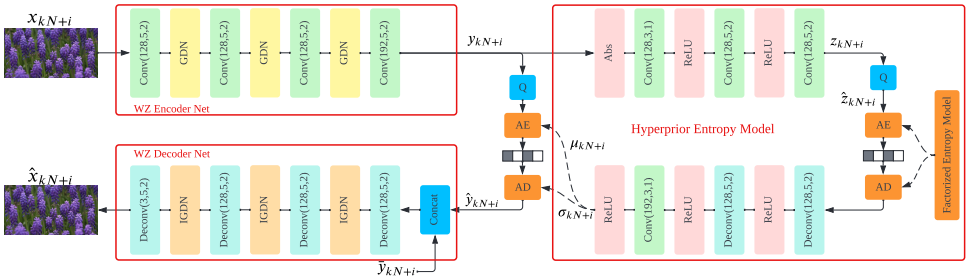


Figure 10: Proposed lite WZ encoder-decoder network.

quantized to $\hat{y}_{kN+i}$ according to predefined quantization levels $\mathcal{Q} = \{0, 2^m | m = 1, ..., 7\}$. The value 0 indicates that some transform bands are not encoded and will be replaced by the SI's corresponding bands at the decoder, while the other bands are divided into multiple bit planes that are processed by next module.

Step 2. Slepian-Wolf encoding. An LDPC accumulate encoder is used to encode the bit planes of $\hat{y}_{kN+i}$ separately and generate the corresponding parity information to be stored in a buffer and sent in chunks upon the request from the decoder via the feedback channel.

Step 3. Side information generation. Based on two previous decoded frames $\hat{x}_{v_1}, \hat{x}_{v_2}$ and the hierarchical frame interpolation order shown in Figure 2(a), a motion compensated frame interpolation algorithm is used to create an SI frame $\bar{x}_{kN+i}$ that is transformed to $\bar{y}_{kN+i}$. The correlation noise between $\hat{y}_{kN+i}$ and $\bar{y}_{kN+i}$ is modeled by a Laplacian distribution as a virtual channel model (Brites & Pereira, 2008). Then the soft information (i.e., conditional bit probabilities P_{cond}) for each bitplane is estimated by using the SI representation $\bar{y}_{kN+i}$, the correlation noise and the previous decoded bit planes.

Step 4. Slepian-Wolf decoding. Given the soft information, each bit plane is decoded by requesting the successive chunks of parity bits from the encoder buffer through the feedback channel until a low bit error probability is achieved.

Step 5. Reconstruction and inverse transformation. The transform bands are firstly reconstructed by grouping the SI's high frequency bands and the decoded bands, followed by de-quantization and inverse transform to obtain the reconstructed WZ frame $\hat{x}_{kN+i}$.

B Network Architecture

ChAR component. As shown in Figure 9, each ChAR block is composed of three modules, including a mean transform module, a scale transform module, and a latent residual prediction module. It is noted that the quantized slice $Q(y_j) = Round(y_j - \mu_j) + \mu_j$ is concatenated with the current slice's mean μ_j and the previous decoded slices $\hat{y}_{<j}$ to obtain the predicted residual r_j. Then the

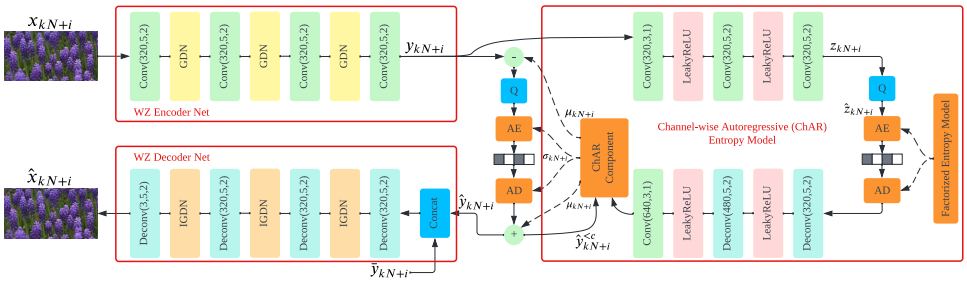
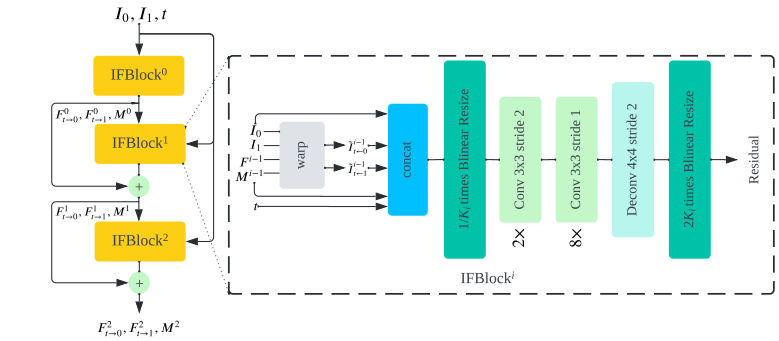
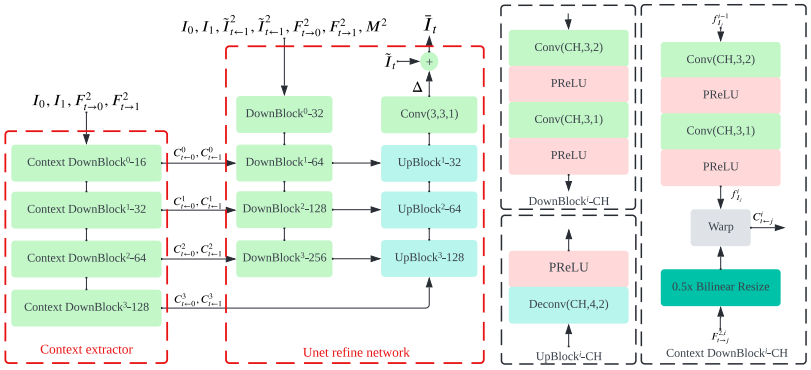


Figure 11: Proposed pro WZ encoder-decoder network.



(a) Network structure of IFNet, where each IFBlock has a resolution parameter, i.e., $(K_0, K_1, K_2) = (4, 2, 1)$.



(b) Network structure of RefineNet with a context extractor and an Unet refine network.

Figure 12: Network structure of RIFE.

current decoded slice $\hat{y}_j$ can be obtained by adding r_j to $Q(y_j)$, hence reducing the quantization error.

Lite WZ encoder-decoder network. Figure 10 illustrates the structures of the proposed lite variant. Compared with the main model, the lite variant uses fewer channels in the convolution layers and only employs the hyperprior entropy model (Ballé et al., 2018b) without the ChAR component.

Pro WZ encoder-decoder network. As shown in Figure 11, the pro variant adopts the same structure with the proposed model but uses more channels. In addition, the number of ChAR blocks is set as 16 to better capture spatial redundancy.

RIFE network. As shown in Figure 12, the RIFE network is composed of an intermediate flow estimation network (IFNet) and a RefineNet. Given two previous decoded frames $\hat{x}_{v_1}, \hat{x}_{v_2}$ and the corresponding time step t $(0 \leq t \leq 1)$, the IFNet estimates the motion information and produces a coarse interpolated frame. Then the RefineNet is used to refine the high-frequency area and reduce

artifacts to create the current predicted frame $\bar{x}_{kN+i}$, which is expected to be as close to the current frame x_{kN+i} as possible. The IFNet adopts several stacked IFBlocks at different resolution to obtain a rough interpolated frame with the following formula:

$$\begin{aligned} \tilde{I}_t &= M \odot \tilde{I}_{t\leftarrow 0} + (1 - M) \odot \tilde{I}_{t\leftarrow 1} \\ \tilde{I}_{t\leftarrow 0} &= \overleftarrow{\mathcal{W}}\left(I_0, F_{t\rightarrow 0}\right), \quad \tilde{I}_{t\leftarrow 1} = \overleftarrow{\mathcal{W}}\left(I_1, F_{t\rightarrow 1}\right) \end{aligned} \tag{2}$$

where M denotes the fusion map ($0 \leq M \leq 1$), the operation $\odot$ is an element-wise multiplier, and $\overleftarrow{\mathcal{W}}$ represents the image backward warping. For each IFBlock, two input frames I_0 and I_1 are first warped to the current frames $\tilde{I}_{t\leftarrow 0}$ and $\tilde{I}_{t\leftarrow 1}$ based on estimated flow F^{i-1} from the $(i-1)^{th}$ IFBlock. Then we concatenate the input frames I_0, I_1, warped frames $\tilde{I}^{i-1}_{t\leftarrow 0}, \tilde{I}^{i-1}_{t\leftarrow 1}$, current timestep t, previous flow F^{i-1} and fusion map M^{i-1} by channel dimension, and feed them into a series of bilinear and convolution operations to approximate the residual of flow and fusion map. After obtaining the the final flow F^2 and the fusion map M^2, we use equation Equation 2 to get the interpolated frame $\tilde{I}_t$.

To refine the high frequency area and reduce the artifacts of $\tilde{I}_t$, the RefineNet is employed to produce a reconstruction residual $\Delta(-1 \leq \Delta \leq 1)$. Specifically, the context extractor first extracts the multi-scale contextual features C_0 and C_1 from input frames I_0 and I_1, respectively. Based on the intermediate flows $F^2_{t\rightarrow 0}$ and $F^2_{t\rightarrow 1}$, these features are warped to the aligned features $C_{t\leftarrow 0}$ and $C_{t\leftarrow 1}$. At the same time, the input frames I_0, I_1, warped frames $\tilde{I}^2_{t\leftarrow 0}, \tilde{I}^2_{t\leftarrow 1}$, the estimated flows $F^2_{t\rightarrow 0}, F^2_{t\rightarrow 1}$ and the fusion map M^2 are concatenated and fed into the encoder of the Unet refine network to produce a refined reconstructed frame $\bar{I}_t = \tilde{I}_t + \Delta$ with the aid of $C_{t\leftarrow 0}$ and $C_{t\leftarrow 1}$.

C EXPERIMENTS

C.1 EXPERIMENTAL DETAILS

Key frame compression. In learning-based video codecs, the pretrained model *mbt-2018* (Minnen et al., 2018) with quality index 4 provided by CompressAI (Bégaint et al., 2020) is used to compress key frames (i.e., the first frame of each GOP). For fair comparison, both DVC [4] (Lu et al., 2019) and DCVC [5] (Li et al., 2021) with the above intra-coding method are retested by using their open source codes. For traditional video codecs, H.264 and H.265 utilize their default intra-coding methods, and WZ video coding (Kodavalla & Mohan, 2012) adopt H.265 intra coding method to compress key frames.

H.264 and H.265 settings. We use the following commands to implement the H.264 and H.265 coding schemes with the only-P mode and the hierarchical-B mode. Specifically, given a sequence *Video.yuv* with the resolution as *W*×*H*, the command lines for generating compressed video using x264 and x265 codecs are as follows:

- H.264(P): *ffmpeg -pix_fmt yuv420p -s:v W×H -i Video.yuv -vframes N_e -c:v libx264 -preset veryfast -tune zerolatency -x264-params "crf=CRF:keyint=GOP:bframes=0:scenecut=0" output.mkv*
- H.264(B): *ffmpeg -pix_fmt yuv420p -s:v W×H -i Video.yuv -vframes N_e -c:v libx264 -preset veryfast -x264-params "crf=CRF:keyint=GOP:scenecut=0:b-adapt=0:bframes=BF:b-pyramid=1" output.mkv*
- H.265(P): *ffmpeg -pix_fmt yuv420p -s:v W×H -i Video.yuv -vframes N_e -c:v libx265 -preset veryfast -tune zerolatency -x265-params "crf=CRF:keyint=GOP:bframes=0" output.mkv*
- H.265(B): *ffmpeg -pix_fmt yuv420p -s:v W×H -i Video.yuv -vframes N_e -c:v libx265 -preset veryfast -x265-params "crf=CRF:keyint=GOP:b-adapt=0:bframes=BF:b-pyramid=1" output.mkv*

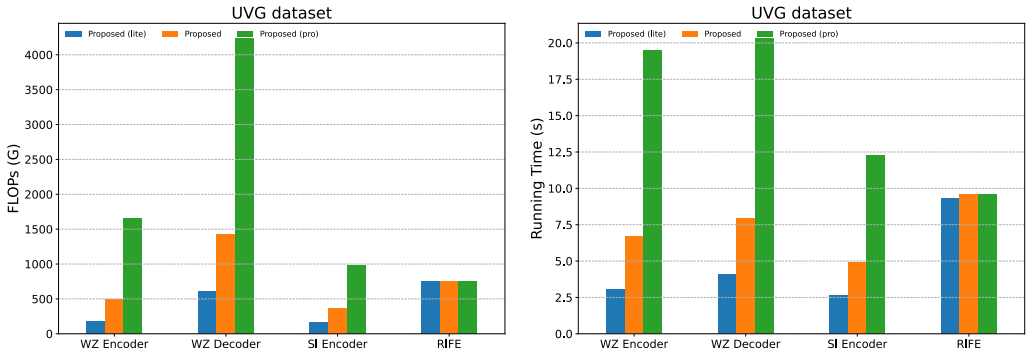
where N_e, CRF, BF represent the number of encoded frames, quantization parameter, and the number of B frames, respectively. Here we set BF as GOP-1 for the hierarchical-B mode.

[4] https://github.com/ZhihaoHu/PyTorchVideoCompression/tree/master/DVC

[5] https://github.com/DeepMC-DCVC/DCVC

Table 4: Decoding complexity of learning-based codecs. H.264 with decoding time as 0.00796s under N_c=1 is set as the anchor to measure BDBR.

Method	FLOPs	Latency					MCL-JCV (BDBR)	
		N_c=1	N_c=2	N_c=4	N_c=8	GPU	PSNR	MS-SSIM
DCVC	3030.68G	63.38s	45.50s	34.30s	27.81s	–	-54.01%	-65.52%
DVC	1317.55G	23.78s	15.16s	10.07s	6.45s	0.226s	-27.95%	-40.45%
DVC-Lite	932.37G	18.31s	10.75s	6.82s	4.25s	0.205s	-13.18%	-35.38%
Proposed (pro)	5979.56G	42.13s	23.93s	15.39s	11.75s	0.306s	-32.18%	-50.15%
Proposed	2542.98G	22.43s	14.09s	9.36s	6.49s	0.190s	-22.63%	-43.04%
Proposed (lite)	1536.36G	16.12s	10.73s	7.03s	4.86s	0.139s	11.36%	-28.34%



(a) (b)

Figure 13: The overall complexity of the proposed methods, including (a) the number of FLOPs and (b) running time for each component on a single CPU core, where the running time of WZ encoder and decoder includes the corresponding entropy coding time.

C.2 Decoder Complexity

Table 4 shows the decoding complexity of six learning-based video codecs on the 1080P test videos. The decoding time is larger than the encoding time, since our proposed methods are based on the distributed coding paradigm, where the decoder performs motion estimation and compensation. The is different from the predictive coding architecture in DVC and DCVC. The results demonstrate that the proposed method not only has lower encoding latency, but also outperforms DVC-Lite with increasing about $22\% \sim 52\%$ decoding time. Moreover, our method achieves the comparable compression performance with similar decoding time to DVC. By adopting a heavier encoder-decoder network, the pro version saves 4.23% (9.7%) bits against DVC in terms of PSNR and MS-SSIM on the MCL-JCV dataset. In addition, although the decoder FLOPs of the pro variant are larger than that of DCVC, the pro version has lower decoding latency. This is explained by the advantage of the ChAR component that can be sped up by parallel computing.

C.3 Complexity Analysis of Each Component

Figure 13 reports the complexity of each component in three different versions of the proposed methods. We observe that the encoder has a lower complexity than the decoder part, which benefits from the advantages brought by the distributed coding architecture. Although each component of the decoder has a large latency on the CPU platform, our proposed method aims at the uplink-based video applications, such as video surveillance and multi-view image acquisition. These applications require a low-power encoder, while the receiver has powerful computation resources to decode the video. As shown in Table 4, the proposed decoder, including the interpolation network, only takes 0.190s to decode a frame of size 1920x1024, which is sufficiently fast to satisfy the basic needs.

C.4 PER-VIDEO LEVEL ANALYSIS

In Figure 14, H.264 with the only-P mode is set as the anchor to compute the BDBR for each video on the UVG and MCL-JCV datasets. We then plot the file size for each compressed video relative to H.264, e.g., a value of 30% in the BDBR represents the relative size as 70% (i.e., 100%-30%).

It is observed that our proposed methods generate smaller encoded files than H.264 and DVC-Lite on most of videos. Compared to DVC, around half of the videos (19/37) compressed by the proposed method have smaller or equal file sizes in terms of PSNR. Nonetheless, for a fraction of videos with fast motion information (e.g., ReadySetGo, video 03, 08 and 12), it is challenging for the proposed model to achieve the superior PSNR performance without utilizing temporal information at the encoder. Moreover, due to the lack of animated videos in the training dataset, our methods cannot generalize well to cartoon videos (i.e., video 18, 20, 24, 25) in the MCL-JCV dataset.

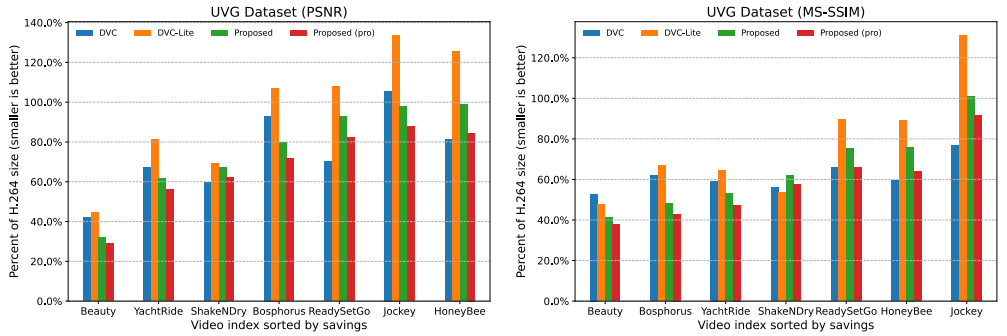
In addition, our methods have better compression efficiency in MS-SSIM than in PSNR, which is partly caused by exploiting SI in the feature space rather than pixel space at the decoder. Thus, the network inclines to focus on structure information instead of pixel information. While there is some performance deficiency under the metric of PSNR in some videos, the proposed Distributed DVC framework is still a feasible attempt to approach the performance of predictive coding.

C.5 FEATURE VISUALIZATIONS

We provide the visualizations of WZ and SI representations in Figure 15 and 16. The SI representation appears to focus on the high-frequency content when compared with the WZ representation. This implies that it is worth exploring how to fully utilize the high-frequency information of SI feature and remove unimportant high-frequency feature maps in the WZ feature to improve the rate-distortion performance in the future.

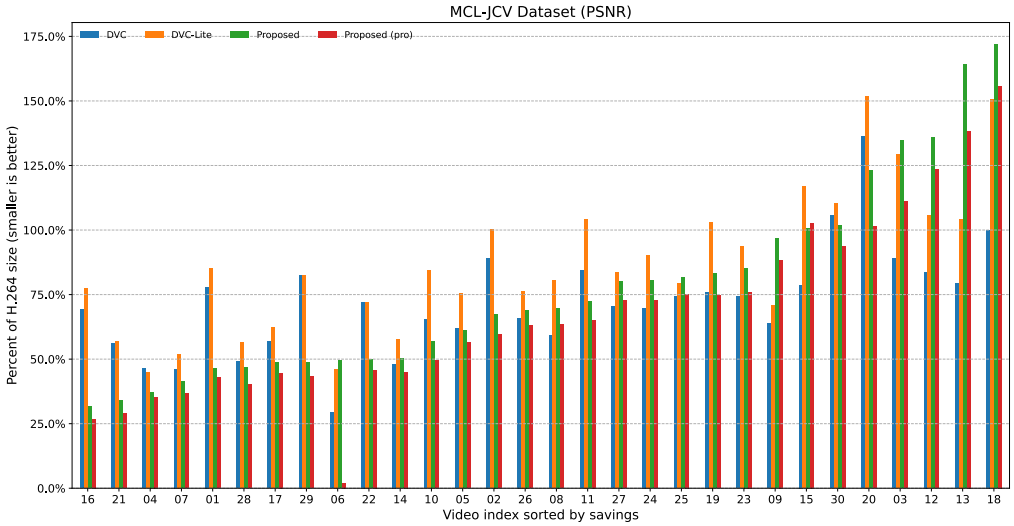
C.6 SUBJECTIVE COMPARISON

In Figure 17, 18, 19 and 20, we show several examples to compare the quantitative results between DVC and our proposed methods. The experimental results verify that our proposed methods achieve better compression performance than DVC in the videos while maintaining plenty of low-motion information. Similar as the results in Figure 14, for some videos with fast motion or cartoon videos, i.e., Figure 19 and 20, our methods have slight performance loss compared with DVC. We believe further performance improvement can be achieved, as discussed in Section 5.

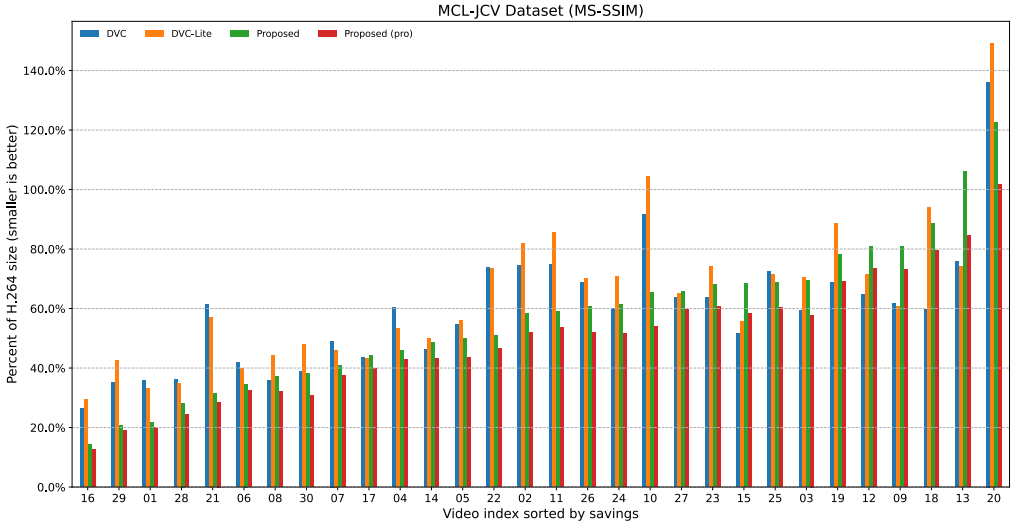


(a)

(b)



(c)



(d)

Figure 14: Rate savings for each video on the UVG and MCL-JCV datasets. Values denote the relative size compared to H.264(P) when measured by BDBR at the same reconstruction level, where the H.264 relative size is always 100%.

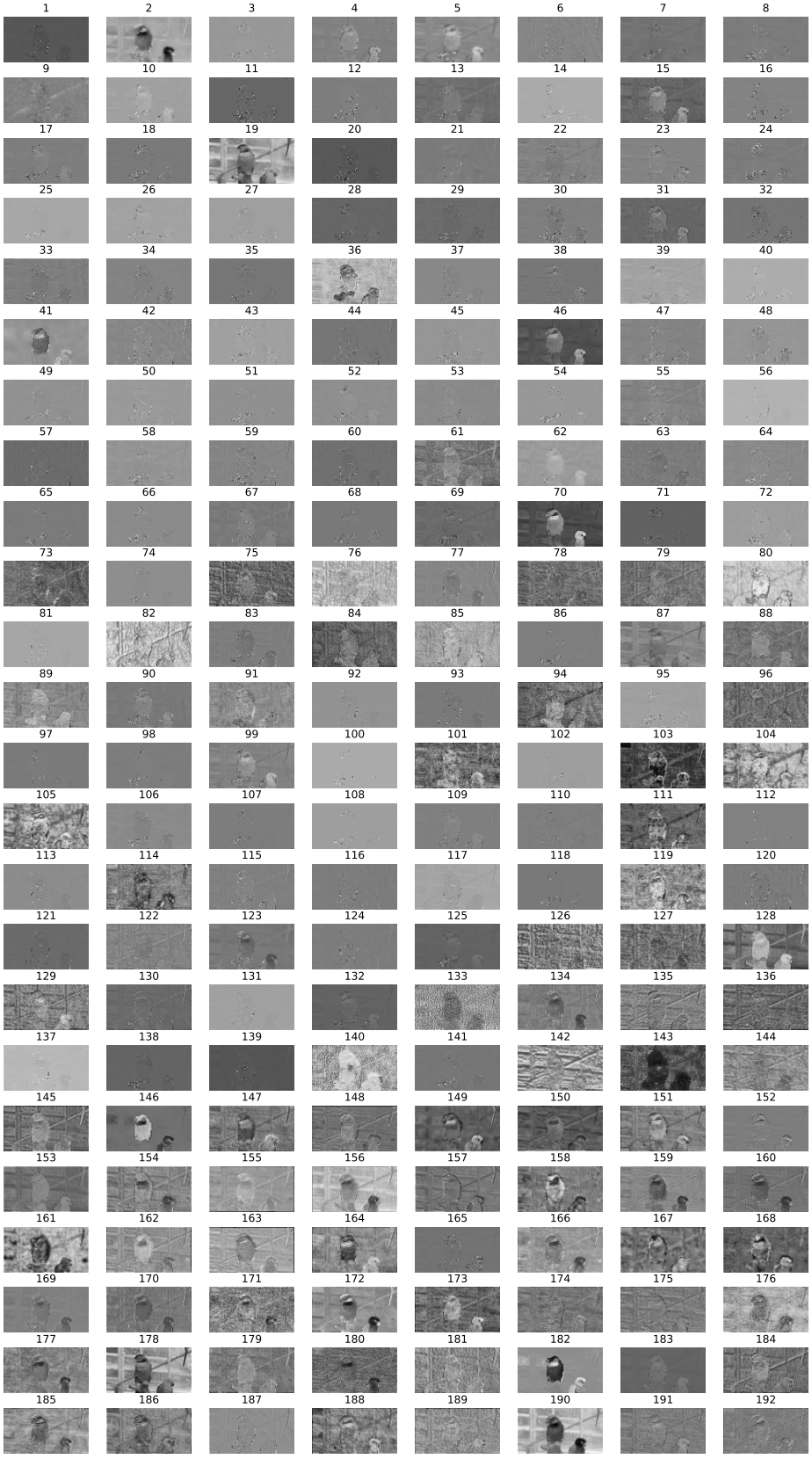


Figure 15: Visualizations of Wyner-Ziv representation from *videoSRC30* sequence in the MCL-JCV dataset.

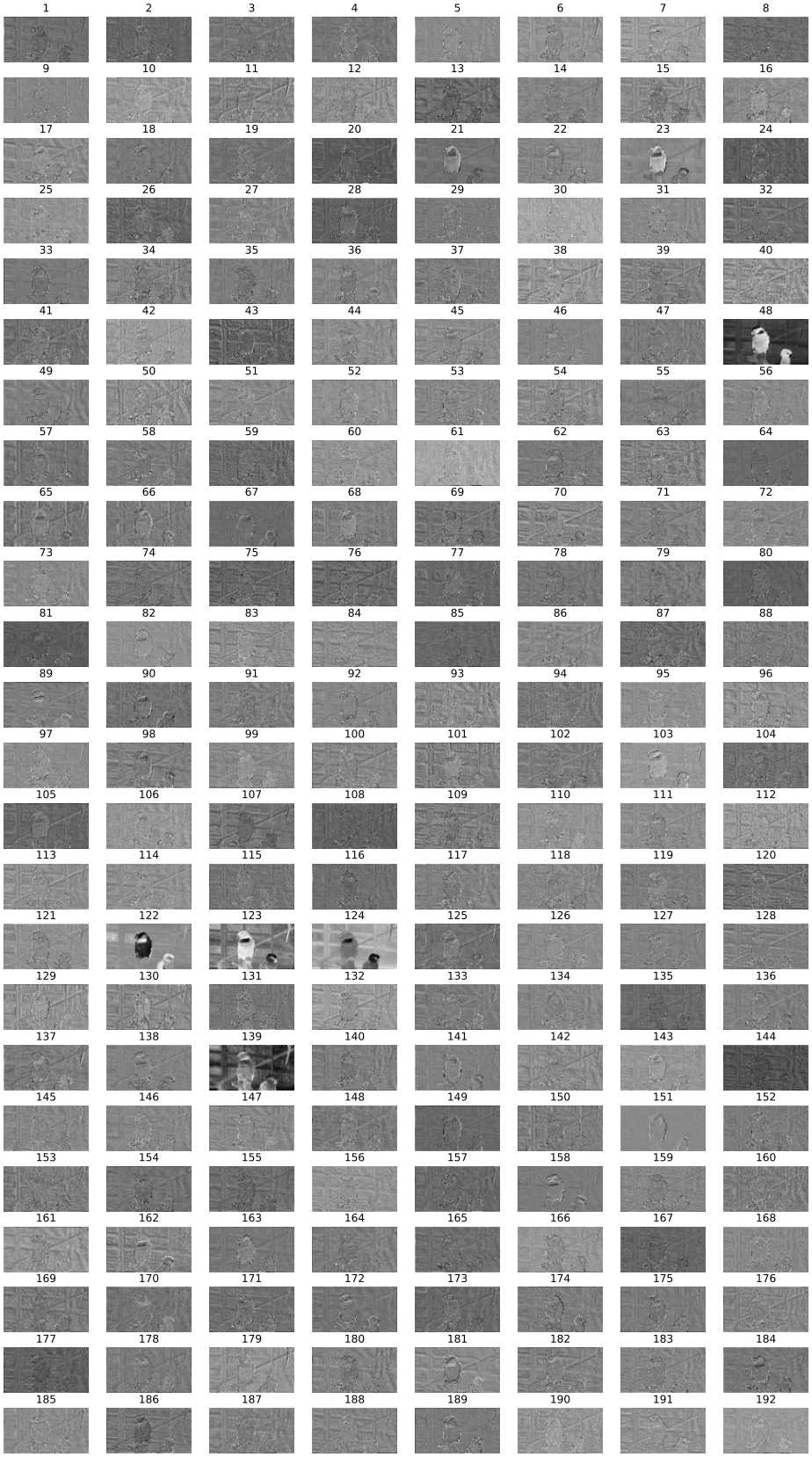
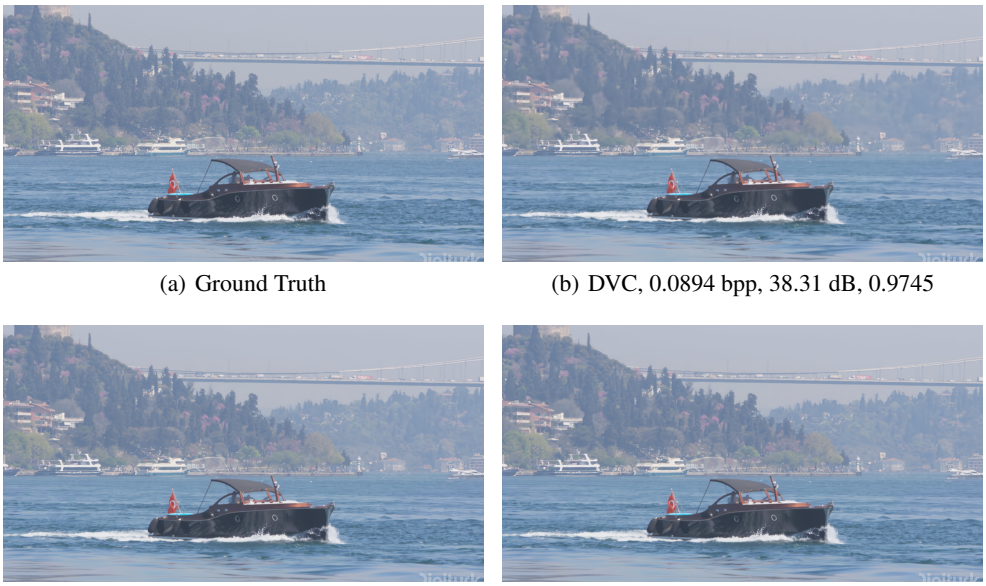


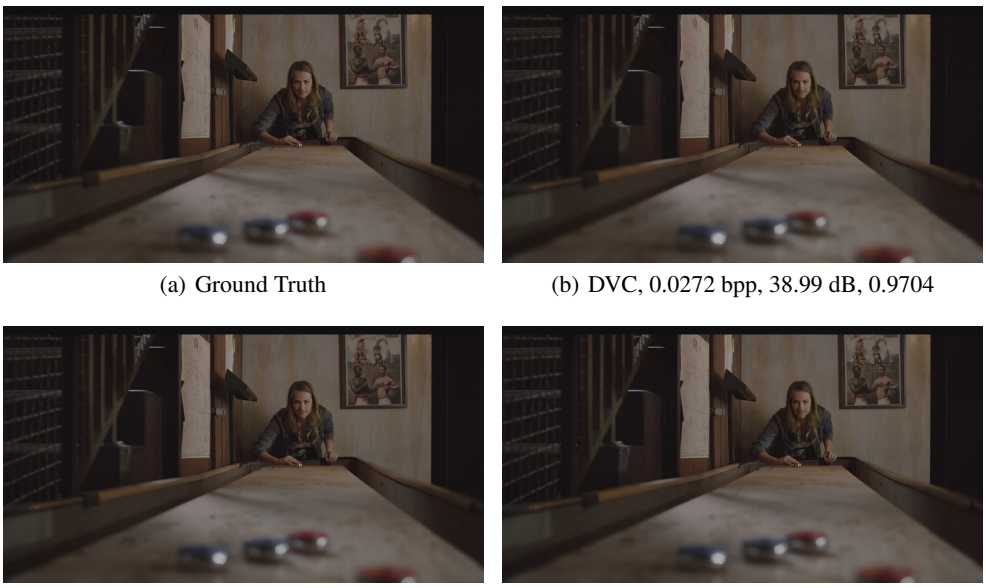
Figure 16: Visualizations of side information representation from *videoSRC30* sequence in the MCL-JCV dataset.



(a) Ground Truth (b) DVC, 0.0894 bpp, 38.31 dB, 0.9745

(c) Proposed, 0.0826 bpp, 38.57 dB, 0.9773 (d) Proposed (pro), 0.0667 bpp, 38.44 dB, 0.9768

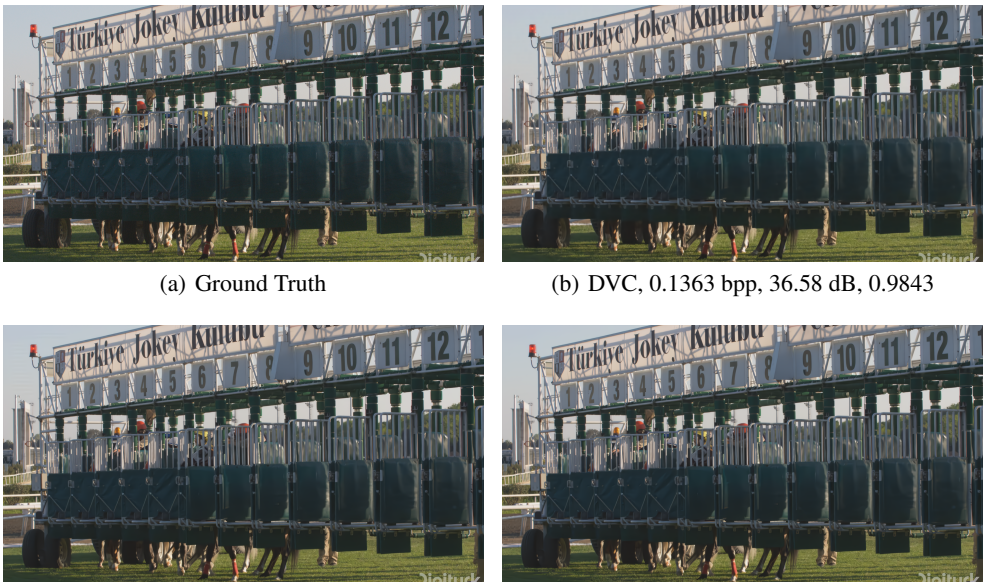
Figure 17: Visual comparison on the *Bosphorus* sequence in the UVG dataset. Our proposed methods save more bits at the same reconstruction quality level.



(a) Ground Truth (b) DVC, 0.0272 bpp, 38.99 dB, 0.9704

(c) Proposed, 0.0208 bpp, 38.98 dB, 0.9718 (d) Proposed (pro), 0.0175 bpp, 39.02 dB, 0.9720

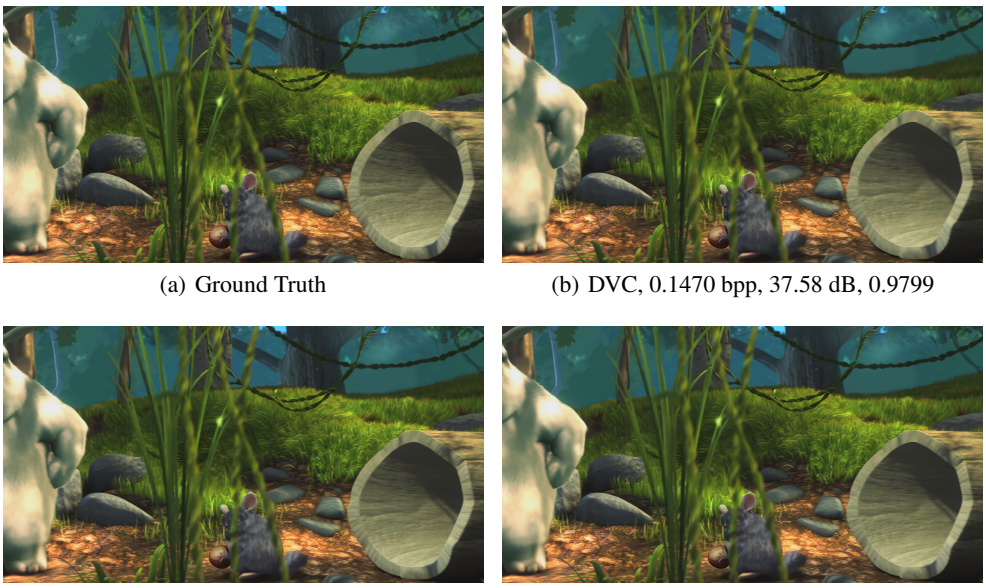
Figure 18: Visual comparison on the *videoSRC01* sequence in the MCL-JCV dataset. Our proposed methods save more bits at the same reconstruction quality level.



(a) Ground Truth (b) DVC, 0.1363 bpp, 36.58 dB, 0.9843

(c) Proposed, 0.1719 bpp, 36.57 dB, 0.9844 (d) Proposed (pro), 0.1405 bpp, 36.48 dB, 0.9843

Figure 19: Visual comparison on the *ReadySetGo* sequence in the UVG dataset. Our proposed methods require more bits at the same reconstruction quality level.



(a) Ground Truth (b) DVC, 0.1470 bpp, 37.58 dB, 0.9799

(c) Proposed, 0.1213 bpp, 37.04 dB, 0.9783 (d) Proposed (pro), 0.1002 bpp, 36.89 dB, 0.9773

Figure 20: Visual comparison on the *videoSRC18* sequence in the MCL-JCV dataset. Our proposed methods require more bits at the same reconstruction quality level.