Parameter-Efficient Instance-Adaptive Neural Video Compression

Hyunmo Yang*,1, Seungjun Oh*,1, and Eunbyung Park†,1,2

[1]Department of Artificial Intelligence, Sungkyunkwan University
[2]Department of Electrical and Computer Engineering, Sungkyunkwan University

Abstract. Learning-based Neural Video Codecs (NVCs) have emerged as a compelling alternative to the standard video codecs, demonstrating promising performance, and simple and easily maintainable pipelines. However, NVCs often fall short of compression performance and occasionally exhibit poor generalization capability due to inference-only compression scheme and their dependence on training data. The instance-adaptive video compression techniques have recently been suggested as a viable solution, fine-tuning the encoder or decoder networks for a particular test instance video. However, fine-tuning all the model parameters incurs high computational costs, increases the bitrates, and often leads to unstable training. In this work, we propose a parameter-efficient instance-adaptive video compression framework. Inspired by the remarkable success of parameter-efficient fine-tuning on large-scale neural network models, we propose to use a lightweight adapter module that can be easily attached to the pretrained NVCs and fine-tuned for test video sequences. The resulting algorithm significantly improves compression performance and reduces the encoding time compared to the existing instant-adaptive video compression algorithms. Furthermore, the suggested fine-tuning method enhances the robustness of the training process, allowing for the proposed method to be widely used in many practical settings. We conducted extensive experiments on various standard benchmark datasets, including UVG, MCL-JVC, and HEVC sequences, and the experimental results have shown a significant improvement in rate-distortion (RD) curves (up to 5 dB PSNR improvements) and BD rates compared to the baselines NVC.

Keywords: Video compression · Instance-adaptation · Parameter-efficient fine-tuning

1 Introduction

In the current digital landscape, we are experiencing unprecedented growth in video content consumption. Despite technological advancements providing us with high-speed internet and significant storage capabilities, efficient video compression technology still remains essential to the whole system. Standard codecs,

* Equal contribution. Authorship order determined by coin flip.
† Corresponding author.

including H.264 [50], H.265 [47], and H.266 [5], have played a critical role in ensuring seamless multimedia experiences.

As an alternative approach to conventional standard codecs, data-driven learning-based video codecs have gained significant attention due to their promising compression performance. Numerous studies have adapted and redesigned deep neural networks to perform encoding and decoding tasks in place of manually crafted algorithms [2, 24, 31, 39]. By leveraging the power of neural networks to autonomously learn efficient representations of video signals and showing the potential of data-driven video codecs, they have sparked considerable interest in further research and development in this field.

The limitations of Neural Video Codecs (NVCs) primarily stem from their reliance on training data. While a network trained on large-scale video datasets may exhibit good performance across a broad spectrum of video types, this assumption does not always hold true in real-world scenarios due to various reasons, such as the lack of diversity in training data, the presence of uncertainty in the optimization processes, and the limited expressibility of the neural networks, among others.

One effective method for enhancing the generalization performance is to further fine-tune the pretrained NVCs on the specific video instance. This approach, known as instance-adaptive fine-tuning, has improved compression performance across different neural codecs. Lu *et al.* [38] suggested fine-tuning only the encoder since modifying the decoder parts requires quantization before sending the data to the receiver. This quantization significantly increases the transmitted bits, resulting in slower framerates at identical bitrates. Despite the drawback associated with decoder fine-tuning, Rozendaal *et al.* [44] demonstrated that a comprehensive fine-tuning approach, encompassing both encoder and decoder, is often superior to encoder-only fine-tuning when only the difference between pretrained and fine-tuned decoder parameters are quantized and transmitted. This strategic approach, applicable to a broad spectrum of learned-based codecs, holds the potential to serve as a pivotal method for enhancing the overall performance of NVCs.

While instance-adaptive methods improve the performance of NVCs, it is noteworthy that achieving this enhancement requires additional codecs (model parameters updated) and encoding time to tailor the model to specific videos. However, fine-tuning the entire model parameters or networks imposes a substantial load on the fine-tuning process, resulting in longer encoding times and an increase in the amount of data bits to be transmitted. In this work, we propose a parameter-efficient instance-adaptive neural video compression method. More specifically, we suggest utilizing the widely recognized Low-Rank Adaptation (LoRA) [22] method to efficiently update the pretrained neural networks. By freezing the parameters of pretrained networks and introducing a few trainable parameters, the proposed instance-adaptive method makes the fine-tuning process more efficient, significantly improving fine-tuning (encoding) time compared to the previous full fine-tuning approaches. Moreover, as only a few parameters are updated during training, the amount of data required for transmission can

be substantially reduced. Furthermore, unlike the full-fine-tuning instance adaptation method, the proposed LoRA-based instant-adaptive strategy exhibits a more stable and robust fine-tuning process.

Among the many well-established neural video codecs, we investigate the effectiveness of the proposed method on SSF [2]. This particular method stands out for its high-quality compression performance and consists of a typical set of video compression modules, such as image compression, flow prediction, and residual compression. Consequently, the comprehensive research conducted in this work can be readily transferred to other neural video codecs. The following are the main contributions of this paper.

- To the best of our knowledge, this work is the first effort to utilize the LoRA type of parameter-efficient fine-tuning method for video compression tasks.
- We have investigated LoRA variants, examining the methods and locations for integrating LoRA modules into well-established neural video codecs.
- The extensive experimental results on many standard benchmark datasets show that the proposed approach has significantly improved the performance compared to the baseline methods.

2 Related Work

2.1 Neural Video Compression

Building upon traditional codecs like H.264 [50] and H.265 [47], DVC [39] introduces a novel architecture that incorporates optical flow for motion compensation and utilizes an encoder-decoder structure composed of convolutional layers. This architecture compresses both residual information and motion derived from optical flow. Numerous subsequent studies have further enhanced this architecture with advanced techniques, including the use of multiple reference frames [25, 35], recurrent auto-encoders and probability models [53], and contextual learning [31–33, 46]. As a well-known way for improving NVCs, motion compensation has been enhanced through the transition from optical flow to scale space flow [2], deformable convolution [25, 52], or cross-scale flow [15].

Despite their advancements, a notable limitation is their poor generalization, resulting from dependence on training data. Considering that training all possible domains is impractical, recent studies have explored the solutions to address this problem. CANF-VC [20] and its subsequent studies [9, 10, 16] leverage augmented normalizing flow. MMVC [36] introduces different modes corresponding to the feature. Among various methods, we select online adaptation to overcome these challenges.

2.2 Content-Adaptive Compression

Neural data compression methods, trained on extensive datasets, can struggle with performance degradation when the data domain differs from the training set or if the data is exceedingly complex. To overcome this constraint, numerous

studies fine-tuned the test data or out-of-domain data. Certain method optimize network without updating decoding parts [1, 6, 13, 14, 54, 55], which has shown promise for model optimization. Additional network application for domain transfer in Neural Image Compression [41, 45, 48] has indicated that fine-tuning with test data domain can enhance compression quality.

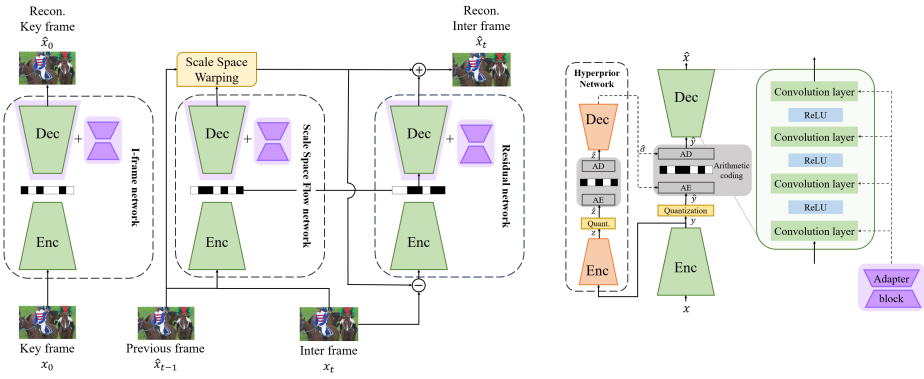
In video compression, Lu *et al*. [38] demonstrated performance improvement by updating only the encoder network, without the need to transmit updated model information. Conversely, some research updates the decoding part. Rozendaal *et al*. [44] employs entire model parameters for overfitting the test data and transmits the changes after training. This approach has resulted in an enhancement of the compression quality compared with their base models. However, the improvement was less significant in low-resolution videos because of the increased bit-rate cost associated with the updated parameters. Research has also been conducted on updating only a portion of the parameters in the decoder network [29, 57], offering a balanced approach between performance improvement and computational efficiency.

Implicit Neural Representation (INR) methods have also attempted to optimize specific videos. NeRV [8] was one of the pioneers in integrating INR into the video compression pipeline. Due to its fast decoding time, NeRV was considered a potential replacement for traditional codecs. However, subsequent studies [7,17,28,30,34,51,56], despite aiming to enhance quality, have shown worse reconstruction performance compared to traditional video codecs and NVC methods.

2.3 Parameter-Efficient Fine Tuning

As models continue to grow in size, the increasing computational costs and insufficient memory storage significantly hinder effective model training. The pioneer of adapters [43] introduces the injection concept to the architecture. Adapter [21], comprised of down projection, up projection, and non-linear layers, is designed to construct a new branch sequentially between the pretrained layers. This sequential integration improves memory efficiency and reduces computation costs. Subsequent studies [11, 12, 37] have expanded the application area of adapters, broadening the adapter architecture. These advances have yielded successful results in areas such as image compression [45, 48].

Despite these advantages, adapters encounter latency issues during inference, primarily due to the presence of non-linear operations. LoRA [22] addresses this concern by eliminating the non-linear layer within the adapter module, outperforming not only adapters but also full fine-tuning in the field of natural language processing (NLP). Numerous LoRA-based researches [19, 26, 40] have successfully extended their applications to various vision areas. Notably, [41] establishes a connection between image compression and LoRA in the decoder. Our novel approach involves integrating the LoRA module into the CNN layer within the decoder of a video codec, a previously unexplored way.



(a) Scale-Space Flow model architecture (b) Compression and adapter insertion scheme

Fig. 1: (a) provides an overview of SSF [2] decoding sequences. The quantized adapter weight must be transferred before decoding the video. (b) illustrates the structure of the compression model using hyperprior network [3]. AE and AD stand for Arithmetic Encoding and Arithmetic Decoding, respectively. The input image x is compressed to the codes y, which are then quantized to $\hat{y}$. The image is subsequently reconstructed to $\hat{x}$ through the decoder, comprising four transposed convolution layers. We have attached an adapter module to each layer of the decoder.

3 Method

3.1 Overview

In this section, we describe the proposed method, parameter-efficient instance-adaptive video compression. We employ the scale-space flow [2] as our baseline model, which compresses the I-frames (Intra-coded frames, or key frames) and P-frames (Predicted frames) through encoder-decoder neural networks. As shown in Fig. 1, This model is characterized by three primary structures, each of which compresses different parts of the video: the key frame, motion information, and residual information. Each encoder-decoder pair uses a hyperprior network for compressing latent information. The proposed method involves the insertion of an adapter into the decoder layers. This allows us to maintain the original model parameters while learning new ones, thereby enhancing each video instance's reconstruction performance. Subsequently, the adapter parameters are transmitted to the receiver side, ensuring that the receiver can access a video of improved quality upon receipt.

3.2 Preliminary: LoRA

LoRA [22] is a parameter-efficient fine-tuning technique for large neural networks. The core idea of LoRA is to train only a few model parameters during fine-tuning, making the fine-tuning more efficient without additional inference latency due to its linear operations. Given the frozen pretrained weight $W_0 \in \mathbb{R}^{C_{out} \times C_{in}}$, where C_{out} and C_{in} are the number of input and output

channels, the trainable weight $\Delta W \in \mathbb{R}^{C_{out} \times C_{in}}$, which is used as an additional parallel branch layer, is introduced to find the optimal weight $W \in \mathbb{R}^{C_{out} \times C_{in}}$ by adjusting only a small subset of the parameters. LoRA assumes that the rank of ΔW is low, and as such, it is composed of a down-projection weight $A \in \mathbb{R}^{r \times C_{in}}$ and an up-projection weight $B \in \mathbb{R}^{C_{out} \times r}$, where $r \ll \texttt{min}(C_{in}, C_{out})$. Consequently, the weight matrix can be reparameterized as follows,

$$W = W_0 + \Delta W = W_0 + AB. \tag{1}$$

Only the low-rank matrices (A and B) are trainable weights and the reparameterization incurs no additional latency during the testing inference.

3.3 Adaptation Modules for Neural Video Codecs

LoRA has been predominantly utilized in transformer architectures and attached to attention and linear layers. Since neural video codecs typically consist of multiple convolutional layers, we developed the revised LoRA architecture to make it compatible with these neural video codecs.

LoRA in convolutional layers At first glance, incorporating the LoRA technique into convolutional layers may not seem challenging. Since convolution is a linear operator (it generally holds associativity and distributivity), we can define a LoRA module with two convolutional filters A and B along with the original convolutional filter W_0 as follows.

$$W_0 * x + B * A * x = (W_0 + B * A) * x = (W_0 + \Delta W) * x, \tag{2}$$

where $*$ is the convolution operator. Similar to how LoRA operates in fully connected layers, it can reduce the number of channels in the first layer (with the filter A), followed by the second convolutional layer (with the filter B) to have the same number of output channels as the original convolutional layer (with the filter W_0). Except for the non-linear activation functions in the middle, it shares similarities with the widely known bottleneck convolution block [18].

However, the convolutions in deep neural networks easily break this assumption in practice due to various reasons, such as discrete convolution, small kernel sizes compared to the input features, and up (or down) samplings. In this work, therefore, we revised a rather simpler technique to efficiently represent the original convolutional kernels.

Factorizing convolution kernels Let $W_0 \in \mathbb{R}^{C_{out} \times C_{in} \times K \times K}$ be a weight tensor for a convolutional layer, where K represents the kernel size. Given the input feature $F_{in} \in \mathbb{R}^{C_{in} \times H \times W}$, where H, W are height and width sizes, a convolutional layer linearly transforms F_{in} into the output feature $F_{out} \in \mathbb{R}^{C_{out} \times H \times W}$. Similar to the original LoRA method, we introduce the trainable weight tensors $A \in \mathbb{R}^{r \times C_{in}}$ and $B \in \mathbb{R}^{C_{out} \times r}$, where $r \ll \texttt{min}(C_{in}, C_{out})$. Note that the number of training parameters is significantly smaller than the original parameters

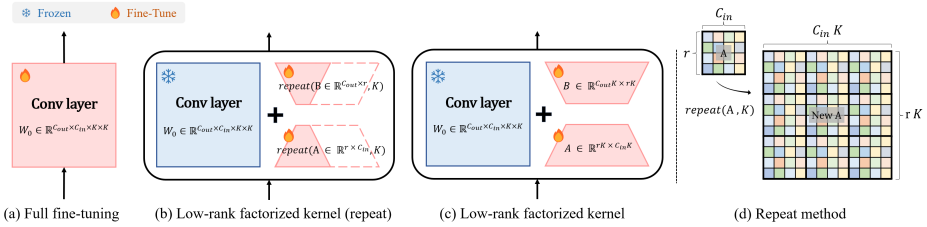


Fig. 2: Illustration of our proposed adapter architecture. (a) represents the original fine-tuning method that updates all parameters within the network. (b) and (c) only update the adapter network, where (c) uses more parameters than (b). (b) duplicates matrices according to the kernel size. (d) illustrates the repeat method applied in (b).

(e.g., $C_{in}, C_{out} = 128, K = 5, r = 8$, we fine-tune only 0.5% of parameters). To merge the original parameters and the newly introduced factorized matrices, we perform matrix multiplication and duplicate the resulting matrix to align its dimensions with those of the original parameters. More formally, the updated weight can be written as,

$$\widehat{A} = \texttt{repeat}(A, K), \tag{3}$$

$$\widehat{B} = \texttt{repeat}(B, K), \tag{4}$$

$$\texttt{repeat}(A, K) = A \otimes J_K, \tag{5}$$

$$W = W_0 + \texttt{reshape}(\widehat{B}\widehat{A}), \tag{6}$$

where $\otimes$ denotes the kronecker product and J_K represents $K \times K$ all-ones matrix. Hence, $\texttt{repeat}(\cdot, \cdot)$ copies the input matrix K^2 times and concatenates the duplicated matrices to construct an enlarged matrix as depicted in Fig. 2-(d), and the repeated matrices have the shapes of $\widehat{A} \in \mathbb{R}^{rK \times C_{in}K}$ and $\widehat{B} \in \mathbb{R}^{C_{out}K \times rK}$. Subsequently, we apply a reshape operator $\texttt{reshape}(\cdot) : \mathbb{R}^{C_{out}K \times C_{in}K} \rightarrow \mathbb{R}^{C_{out} \times C_{in} \times K \times K}$ to construct a ΔW to be added to the original convolutional kernel W_0.

Despite its minimal parameter usage, the proposed factorization approach is remarkably efficient for compressing videos on a per-instance basis. However, the small number of update parameters sometimes results in limited performance improvements on some datasets and faces challenges in scenarios requiring high bitrate compression. Using a larger value for r can easily increase the number of trainable parameters, but through empirical observation, we have noticed that merely raising the rank often does not improve the fine-tuning process.

The additional structure we propose largely mirrors the previous approach, with the key difference being that it does not duplicate the matrix but instead uses a larger number of parameters. With the slight abuse of notation, this method involves two matrices that decompose the weight of the convolution W_0 into $A \in \mathbb{R}^{rK \times C_{in}K}$ and $B \in \mathbb{R}^{C_{out}K \times rK}$, where K continues to represent the kernel size, consistent with the previous structure. The remaining elements of the structure are configured similarly to the previous setup. The updated parameter

W can be written as,

$$W = W_0 + \texttt{reshape}(BA). \tag{7}$$

The proposed method starts to adjust the adapter parameters for a video instance and freeze the pretrained parameters. By initializing the adapter parameters to zero, the model replicates the output of the original model at the beginning of the fine-tuning process.

3.4 Instance-Adaptive fine-tuning

Given a video instance during testing time, the proposed method performs fine-tuning by updating the proposed factorized convolutional kernels. After a few training iterations, the updated weights are quantized and compressed before transmission to the receiver. To maximize the compression ratio, they are also entropy-coded to the bitstream with a prior, along with the latent codes generated by the encoder. On the receiver side, it already has the pretrained decoder parameters and updates the convolutional kernels. The resulting model architecture remains identical to the pretrained model, hence no additional latency during decoding.

Decoder-only updates Modern NVCs have utilized the encoder and decoder architectures, where the encoder extracts the codes from the input videos, and the spatial resolution is downsampled along the feature extraction process. On the other hand, the decoder upsamples the extracted codes to reconstruct the videos up to the original resolution. While it is possible to fine-tune both the encoder and decoder, our empirical observations indicate that competitive performance can be achieved by fine-tuning only the decoder. Fine-tuning the encoder results in a slight improvement in the compression ratio, but it requires a longer training time due to the need for backpropagation operations down to the input layer. Similar results have been demonstrated by Rozendaal *et al.* [44], showing the limitations of encoder updates in the case of full-fine-tuning. We will provide experimental results in the Sec. 4.3.

Vanishing parameters In the fine-tuning stage, we observed that a significant amount of update parameters are vanishing in the quantization process. This results in substantial performance degradation on the receiver side. The primary cause of this issue was the slight modifications made to the parameters, which can be lost during quantization by falling into the zero bin. To prevent significant information loss, we use higher learning rates to encourage larger updates to the parameters. This enhances compression performance and significantly reduces training time since we achieve good reconstruction quality in fewer training iterations.

Rate-distortion fine-tuning The training loss optimized during the training is given by the following equation:

$$L = \frac{1}{N}\sum_{i=0}^{N-1} \lambda D(x_i, \hat{x}_i) + H(z_i) + \beta H(\bar{w}), \tag{8}$$

where λ is trade off between D and H, D denotes distortion loss between ground truth x and reconstructed data $\hat{x}$, and H is entropy estimation to represent the compressed latent information from I-frame, motion, and residual. As D can be either mean square error (MSE) or structural similarity index measure (SSIM), we use D as MSE loss. $H(\bar{w})$ with coefficient β represents the number of bits of the quantized factorized kernel weights. Following Rozendaal *et al.* [44], we set spike-and-slab prior [27] to estimate the entropy estimation of the factorized kernel weights.

$$p(w) = \frac{\mathcal{N}(w|0, \sigma^2 I) + \alpha \mathcal{N}(w|0, s^2 I)}{1 + \alpha}, \tag{9}$$

where σ^2 and s^2 denotes the variances of slab and spike components, respectively, and α is tunable parameter to set the scale of spike prior. The slap component keeps a lower scale of updates, while spike makes zero-update, enabling cheaper and sparser updates.

4 Experiments

4.1 Experimental Setup

Dataset We evaluate the performance of our method on the UVG-1k dataset [42], MCL-JCV dataset [49], and the HEVC class B and C dataset [4]. We use RGB format for all video sequences. The resolution of the UVG dataset, MCL-JCV dataset, and the HEVC class B dataset are 1920×1080, with 7, 30, and 5 videos in each dataset, respectively. Due to our backbone model, which requires input sizes with a height and width that are multiples of 128, we crop the video resolution to 1920×1024. The HEVC class C dataset consists of 4 videos with a resolution of 832×480. For this dataset, We pad the right and bottom sides of the input image to fit into the model and crop outputs to obtain the final video.

Video adaptation We train each video using an MSE-optimized pretrained SSF model [2]. We use nine qualities, each trained along to bitrate. We set λ as $0.01 \cdot 2^i$, where i ranges from -3 to 5, following the original approach. Both full fine-tuning and adapter fine-tuning methods are applied to train our model. We set α to 1000 and β to 1 for loss calculation. The baseline model has 4 layers on the decoder, and we set the rank of the adapter r as 16, 8, 8, and 2, starting from the first layer. Regardless of the frame length of the video data, we conducted 15 epochs of training for all video sequences. For the learning rate, we use 0.0001 for full fine-tuning and 0.0005 for our approach. We observed that using a high learning rate for full fine-tuning led to poorer results as training progressed, so

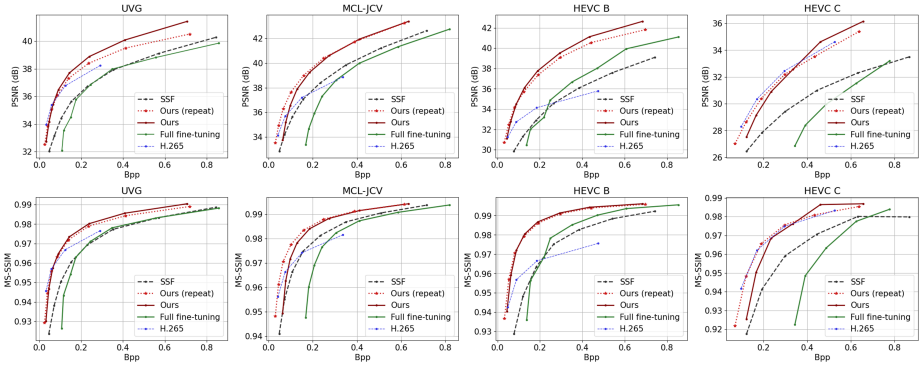


Fig. 3: Rate-Distortion curve comparison with the baseline method, SSF [2] on UVG, MCL-JCV, HEVC class B, and C datasets

Method	UVG	MCL-JCV	HEVC B	HEVC C	Avg.
SSF	111.26	22.40	50.63	121.10	76.35
Full fine-tuning	137.03	101.11	52.99	167.35	114.62
Ours (repeat)	-0.14	**-35.85**	**-50.42**	**9.26**	**-19.29**
Ours	**-6.48**	-11.14	-47.47	14.78	-12.58

Table 1: BD-rate (%) comparison with x265. A lower value indicates better performance compared to the reference codec.

we selected a value that could stably improve performance. Additionally, we use the 'ReduceOnPlateau' learning rate scheduler. Since we use a higher learning rate than that used in training and train on a single video, once the model quickly converges, we reduce the learning rate to allow further training progress. We set the Group of Pictures (GoP) to 4 during training, with batch size 3. For testing, we set the GoP to 12. Previous study [2] have reported that training with smaller GoPs can lead to quicker convergence, and this strategy was also applied to instance-adaptation. Using smaller GoPs, rather than learning the testing sequences as they are, was beneficial for rapid convergence.

4.2 Experimental Result

Quantitative Results We evaluate the performance with Rate-distortion (RD) curve and BD-rate, anchoring on x265. The data for this codec was obtained from publicly available data online [23]. Fig. 3 shows the RD curve measured for PSNR and MS-SSIM on the UVG, MCL-JCV, and HEVC class B and C datasets. Both methods we proposed significantly improved the performance of the existing model. While the instance adaptive method shows relatively weak

Fig. 4: Qualitative result for moving object, numbers in the pictures represent the corresponding bpp. Our method effectively reduces distortions in P frames, resulting in a clearer and more accurate representation in similar bpp. (Left) 'BasketballDrive' sequence from the HEVC class B dataset. (Right) 'ReadySteadyGo' sequence from the UVG dataset.

performance on the low resolution data [44], our proposed methods also demonstrate similar performance improvement on the HEVC class C dataset, which has smaller resolution compared to other datasets.

Tab. 1 presents the BD-rate results. We also conducted tests on the UVG, MCL-JCV, and HEVC class B and C datasets. Our method shows significant improvement compared to the original result, and the result on the 1K dataset outperformed H.265. Remarkably, the BD-rate of ours with duplication is the best due to significantly fewer trainable parameters, resulting in a lower size of sending bits.

Qualitative Results Fig. 4 presents the video compression results. We compared the reconstructed frames with similar bpp. The fine-tuning method has shown effective adaptability to each video sequence. It was observed that by adopting the instance-adaptive method, we could achieve outputs that closely resemble the original. When compared to the original, significant improvements were observed with reduced motion blur, color distortion, and other artifacts in moving objects. However, performance improvements were not noticeable in full fine-tuning. On the left of the Fig. 4, it is evident that the SSF method distorts color and retains motion information. In contrast, these errors are mitigated in the instance-adaptive method. Notably, our method reduces degradations, particularly on the ball and human face.

4.3 Ablation studies

Encoder adaptation Encoder-only updates exhibit limited improvement, as discussed in Sec. 3.4 and Rozendaal *et al.* [44]. We performed an experiment involving attachment to both the encoder and decoder, similar to full fine-tuning but with a smaller number of updating parameters. As shown in Tab. 2, our

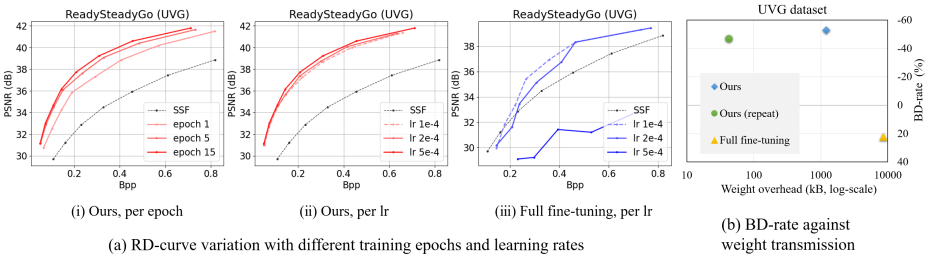
Methods	BD-rate (%)							Training time (min)
	Beauty	ReadySteadyGo	Bosphorus	HoneyBee	Jockey	ShakeNDry	YachtRide	
Full fine-tuning	91.21	-13.89	39.98	126.56	41.05	59.36	15.50	23
Ours (repeat)	-13.89	-62.99	-64.50	-28.68	-41.19	-14.95	-52.85	14
Ours	-28.53	-67.52	-64.33	-47.66	-45.84	-16.33	-56.92	14
Ours (enc, dec)	-34.44	-66.74	-67.71	-53.17	-50.38	-21.77	-59.00	16

Table 2: BD-rate (%) of each data in UVG dataset (with SSF as the anchor), and training time (in Minutes) of one video of 1920×1024 resolution and 600 frames.

method using both encoder and decoder without duplication shows lower BD-rates and faster training time compared to full fine-tuning, similar with using only decoder side. However, attaching both sides has a slight gap and longer training time compared to decoder-only training, leading us to opt for training only the decoder part.

Instance-adaptive environments We conducted an ablation study on various learning rates using ReadySteadyGo, as depicted in Fig. 5. (i) We assessed the convergence speed in our predefined setting. After just one epoch, the curve exhibits a notable difference from the baseline, and the interval progressively narrows as epochs progress, signifies a fast convergence speed. (ii) As mentioned in Sec. 3.4 concerning gradient vanishing, we trained our method with a learning rate ranging from 0.0001 to 0.0005. A slight expansion in the RD curve is observed, considered as quantization loss, which impedes faster convergence. (ii, iii) The results of our proposed method and full fine-tuning across various learning rates are presented. RD curves of full fine-tuning notably decline as the learning rate is increased. This suggests that full fine-tuning requires training at a lower learning rate, taking a longer time for convergence. On the other hand, using only factorized kernel during fine-tuning maintains performance regardless of the learning rate, and achieves this in a shorter time. Therefore, our proposed method demonstrates robustness across various hyperparameter environments.

Additional decoder bits Before using the video codec, it is necessary to update fine-tuned information to synchronize the transmitter and receiver side. After this overhead transmission step is completed, video compression can be performed as in a typical NVC. Fig. 5 (b) demonstrates the quantized decoder overhead with BD-rate. Full fine-tuning requires transmitting 10 times or more weights compared to our proposed methods, despite the increased BD-rate. The repeated method, with significantly fewer trainable parameters, exhibits lower weight overheads and a reduced size of total bits. This is due to its reduced weight to be transmitted, resulting in a decreased total number of bitstreams. On the other hand, the non-repeated method shows the highest BD-rate among the three methods, even though the transmitted weight ratio relative to the overall bitstreams is larger than the repeated method. These results imply that



(i) Ours, per epoch (ii) Ours, per lr (iii) Full fine-tuning, per lr

(a) RD-curve variation with different training epochs and learning rates

(b) BD-rate against weight transmission

Fig. 5: (a): RD curve varies according to training parameters, as measured on the ReadySteadyGo Dataset. (b): BD-rate (with the anchor as SSF) per adapter weight needed for decoding a video. In (a)(i), our method demonstrates that performance gains can be achieved with just one training epoch. (a)(ii) shows that using a higher learning rate can lead to a better PSNR. However, graph (a)(iii) suggests that a high learning rate is not feasible during full fine-tuning. In (b), while the BD-rate of Ours(repeat) and Ours is quite similar, Ours(repeat) requires significantly less overhead at only 43kB, compared to Ours which requires 1025kB, and full fine-tuning which necessitates 8522kB.

our methods require a smaller weight bit size for transmission while achieving superior video compression.

Hard-to-compress data Agustsson *et al*. [2] acknowledged a failure to produce reasonable results on animation style datasets, which are out-of-domain. Fig. 6 illustrates the size of encoded data relative to H.265, indicating the proportion of data used to compress a video with an equivalent PSNR. We selected datasets which yield worse result compared to the traditional codec. Our proposed method has demonstrated its effectiveness in reducing the bitrate required for compression. Therefore, our methods enhance the motion estimation performance and exhibit better generalization across various video domains.

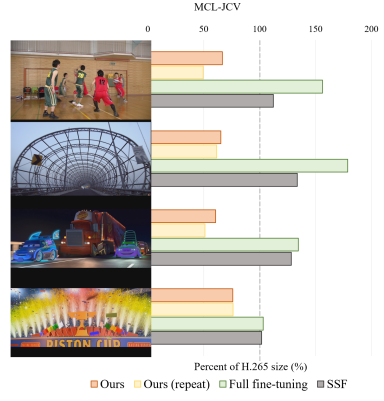


Fig. 6: Rate savings for out-of-domain data. Y axis represents the bitrate compared to H.265(smaller is better).

Apply to another NVC We applied our methods to another baseline, FVC [25], an end-to-end video codec. FVC only compresses P-frames, using H.265 for I-frame compression. As shown in Fig. 7, our method outperforms the baseline in benchmark datasets. This demonstrates that our proposed approach performs well even in different architectures, and also can be applied to P-frame settings.

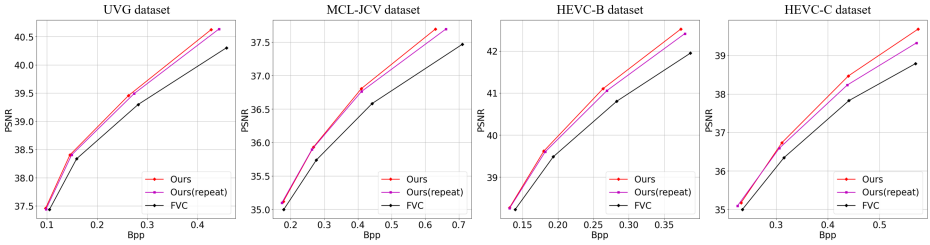


Fig. 7: RD-curve comparison using FVC.

	Ours	Ours (repeat)	Bias-tuning	Tsubota *et al.*	Shen *et al.*
BD-rate (%)	-6.48	-0.14	65.84	9.47	11.69

Table 3: BD-rate comparison with previous methods using anchor as H.265

Comparison to the previous methods We did comparisons to the existing methods applied to other tasks, such as domain transfer and image compression. Tab. 3 represents the BD-rate on UVG dataset compared to H.265 with bias-tuning, Tsubota *et al.* [48](adapter using matrix decomposition) and Shen *et al.* [45](adapter using depthwise separable convolution and activation function). We implemented their methods on video compression tasks, and our methods show the best results among the other PEFT methods. This supports the effectiveness of the proposed architectural design.

5 Conclusion

We introduce a novel parameter-efficient instance-adaptive method, which is adapted to scale-space flow models in both I-frame and P-frame settings. Our approach utilizes linear operations for reparameterization, ensuring no additional latency during decoding. Training factorized kernels with duplications maintain performance while reducing the number of bits transmitted in lower bpp areas, as evidenced by various BD-rate measurements. Training these kernels without duplications yields superior performance in RD-curves. Additional experimental results demonstrate the robustness, speed, and generalization capabilities of our methods. We believe that our work represents a significant step towards enhancing the efficiency and adaptability of instance-adaptive video compression.

References

1. Abdoli, M., Clare, G., Henry, F.: Gop-based latent refinement for learned video coding. In: ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 1–5. IEEE (2023)

2. Agustsson, E., Minnen, D., Johnston, N., Balle, J., Hwang, S.J., Toderici, G.: Scale-space flow for end-to-end optimized video compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8503–8512 (2020)
3. Ballé, J., Minnen, D., Singh, S., Hwang, S.J., Johnston, N.: Variational image compression with a scale hyperprior. arXiv preprint arXiv:1802.01436 (2018)
4. Bossen, F., et al.: Common test conditions and software reference configurations. JCTVC-L1100 **12**(7), 1 (2013)
5. Bross, B., Wang, Y.K., Ye, Y., Liu, S., Chen, J., Sullivan, G.J., Ohm, J.R.: Overview of the versatile video coding (vvc) standard and its applications. IEEE Transactions on Circuits and Systems for Video Technology **31**(10), 3736–3764 (2021)
6. Campos, J., Meierhans, S., Djelouah, A., Schroers, C.: Content adaptive optimization for neural image compression. arXiv preprint arXiv:1906.01223 (2019)
7. Chen, H., Gwilliam, M., Lim, S.N., Shrivastava, A.: Hnerv: A hybrid neural representation for videos. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10270–10279 (2023)
8. Chen, H., He, B., Wang, H., Ren, Y., Lim, S.N., Shrivastava, A.: Nerv: Neural representations for videos. Advances in Neural Information Processing Systems **34**, 21557–21568 (2021)
9. Chen, M.J., Chen, Y.H., Peng, W.H.: B-canf: Adaptive b-frame coding with conditional augmented normalizing flows. IEEE Transactions on Circuits and Systems for Video Technology (2023)
10. Chen, P.Y., Peng, W.H.: Canf-vc++: Enhancing conditional augmented normalizing flows for video compression with advanced techniques. arXiv preprint arXiv:2309.05382 (2023)
11. Chen, S., Ge, C., Tong, Z., Wang, J., Song, Y., Wang, J., Luo, P.: Adaptformer: Adapting vision transformers for scalable visual recognition. Advances in Neural Information Processing Systems **35**, 16664–16678 (2022)
12. Chen, Z., Duan, Y., Wang, W., He, J., Lu, T., Dai, J., Qiao, Y.: Vision transformer adapter for dense predictions. arXiv preprint arXiv:2205.08534 (2022)
13. Djelouah, J., Schroers, C.: Content adaptive optimization for neural image compression. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2019)
14. Gao, C., Xu, T., He, D., Wang, Y., Qin, H.: Flexible neural image compression via code editing. Advances in Neural Information Processing Systems **35**, 12184–12196 (2022)
15. Guo, Z., Feng, R., Zhang, Z., Jin, X., Chen, Z.: Learning cross-scale prediction for efficient neural video compression. arXiv e-prints pp. arXiv–2112 (2021)
16. Hadizadeh, H., Bajić, I.V.: Lccm-vc: Learned conditional coding modes for video coding. arXiv preprint arXiv:2210.15883 (2022)
17. He, B., Yang, X., Wang, H., Wu, Z., Chen, H., Huang, S., Ren, Y., Lim, S.N., Shrivastava, A.: Towards scalable neural representation for diverse videos. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6132–6142 (2023)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2026)
19. He, X., Li, C., Zhang, P., Yang, J., Wang, X.E.: Parameter-efficient model adaptation for vision transformers. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 817–825 (2023)

20. Ho, Y.H., Chang, C.P., Chen, P.Y., Gnutti, A., Peng, W.H.: Canf-vc: Conditional augmented normalizing flows for video compression. In: European Conference on Computer Vision. pp. 207–223. Springer (2022)
21. Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., Gelly, S.: Parameter-efficient transfer learning for nlp. In: International Conference on Machine Learning. pp. 2790–2799. PMLR (2019)
22. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021)
23. Hu, Z.: Pytorchvideocompression. https://github.com/ZhihaoHu/PyTorchVideo Compression (2020)
24. Hu, Z., Lu, G., Guo, J., Liu, S., Jiang, W., Xu, D.: Coarse-to-fine deep video coding with hyperprior-guided mode prediction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5921–5930 (2022)
25. Hu, Z., Lu, G., Xu, D.: Fvc: A new framework towards deep video compression in feature space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1502–1511 (2021)
26. Jie, S., Deng, Z.H.: Fact: Factor-tuning for lightweight adaptation on vision transformer. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 1060–1068 (2023)
27. Johnstone, I.M., Titterington, D.M.: Statistical challenges of high-dimensional data (2009)
28. Kwan, H.M., Gao, G., Zhang, F., Gower, A., Bull, D.: Hinerv: Video compression with hierarchical encoding-based neural representation. Advances in Neural Information Processing Systems **36** (2024)
29. Lam, Y.H., Zare, A., Cricri, F., Lainema, J., Hannuksela, M.M.: Efficient adaptation of neural network filter for video compression. In: Proceedings of the 28th ACM International Conference on Multimedia. pp. 358–366 (2020)
30. Lee, J.C., Rho, D., Ko, J.H., Park, E.: Ffnerv: Flow-guided frame-wise neural representations for videos. In: Proceedings of the 31st ACM International Conference on Multimedia. pp. 7859–7870 (2023)
31. Li, J., Li, B., Lu, Y.: Deep contextual video compression. Advances in Neural Information Processing Systems **34**, 18114–18125 (2021)
32. Li, J., Li, B., Lu, Y.: Hybrid spatial-temporal entropy modelling for neural video compression. In: Proceedings of the 30th ACM International Conference on Multimedia. pp. 1503–1511 (2022)
33. Li, J., Li, B., Lu, Y.: Neural video compression with diverse contexts. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 22616–22626 (2023)
34. Li, Z., Wang, M., Pi, H., Xu, K., Mei, J., Liu, Y.: E-nerv: Expedite neural video representation with disentangled spatial-temporal context. In: European Conference on Computer Vision. pp. 267–284. Springer (2022)
35. Lin, J., Liu, D., Li, H., Wu, F.: M-lvc: Multiple frames prediction for learned video compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3546–3554 (2020)
36. Liu, B., Chen, Y., Machineni, R.C., Liu, S., Kim, H.S.: Mmvc: Learned multi-mode video compression with block-based prediction mode selection and density-adaptive entropy coding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18487–18496 (2023)

37. Liu, Y.C., Ma, C.Y., Tian, J., He, Z., Kira, Z.: Polyhistor: Parameter-efficient multi-task adaptation for dense vision tasks. Advances in Neural Information Processing Systems **35**, 36889–36901 (2022)
38. Lu, G., Cai, C., Zhang, X., Chen, L., Ouyang, W., Xu, D., Gao, Z.: Content adaptive and error propagation aware deep video compression. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16. pp. 456–472. Springer (2020)
39. Lu, G., Ouyang, W., Xu, D., Zhang, X., Cai, C., Gao, Z.: Dvc: An end-to-end deep video compression framework. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11006–11015 (2019)
40. Luo, G., Huang, M., Zhou, Y., Sun, X., Jiang, G., Wang, Z., Ji, R.: Towards efficient visual adaption via structural re-parameterization. arXiv preprint arXiv:2302.08106 (2023)
41. Lv, Y., Xiang, J., Zhang, J., Yang, W., Han, X., Yang, W.: Dynamic low-rank instance adaptation for universal neural image compression. In: Proceedings of the 31st ACM International Conference on Multimedia. pp. 632–642 (2023)
42. Mercat, A., Viitanen, M., Vanne, J.: Uvg dataset: 50/120fps 4k sequences for video codec analysis and development. In: Proceedings of the 11th ACM Multimedia Systems Conference. pp. 297–302 (2020)
43. Rebuffi, S.A., Bilen, H., Vedaldi, A.: Learning multiple visual domains with residual adapters. Advances in neural information processing systems **30** (2017)
44. van Rozendaal, T., Brehmer, J., Zhang, Y., Pourreza, R., Wiggers, A., Cohen, T.S.: Instance-adaptive video compression: Improving neural codecs by training on the test set. arXiv preprint arXiv:2111.10302 (2021)
45. Shen, S., Yue, H., Yang, J.: Dec-adapter: Exploring efficient decoder-side adapter for bridging screen content and natural image compression. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12887–12896 (2023)
46. Sheng, X., Li, J., Li, B., Li, L., Liu, D., Lu, Y.: Temporal context mining for learned video compression. IEEE Transactions on Multimedia (2022)
47. Sullivan, G.J., Ohm, J.R., Han, W.J., Wiegand, T.: Overview of the high efficiency video coding (hevc) standard. IEEE Transactions on circuits and systems for video technology **22**(12), 1649–1668 (2012)
48. Tsubota, K., Akutsu, H., Aizawa, K.: Universal deep image compression via content-adaptive optimization with adapters. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 2529–2538 (2023)
49. Wang, H., Gan, W., Hu, S., Lin, J.Y., Jin, L., Song, L., Wang, P., Katsavounidis, I., Aaron, A., Kuo, C.C.J.: Mcl-jcv: a jnd-based h. 264/avc video quality assessment dataset. In: 2016 IEEE international conference on image processing (ICIP). pp. 1509–1513. IEEE (2016)
50. Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A.: Overview of the h. 264/avc video coding standard. IEEE Transactions on circuits and systems for video technology **13**(7), 560–576 (2003)
51. Xu, Y., Feng, X., Qin, F., Ge, R., Peng, Y., Wang, C.: Vq-nerv: A vector quantized neural representation for videos. arXiv preprint arXiv:2403.12401 (2024)
52. Yang, J., Yang, C., Xiong, F., Wang, F., Wang, R.: Learned low bitrate video compression with space-time super-resolution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1786–1790 (2022)
53. Yang, R., Mentzer, F., Van Gool, L., Timofte, R.: Learning for video compression with recurrent auto-encoder and recurrent probability model. IEEE Journal of Selected Topics in Signal Processing **15**(2), 388–401 (2020)

54. Yang, Y., Bamler, R., Mandt, S.: Improving inference for neural image compression. Advances in Neural Information Processing Systems **33**, 573–584 (2020)
55. Zhao, J., Li, B., Li, J., Xiong, R., Lu, Y.: A universal encoder rate distortion optimization framework for learned compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1880–1884 (2021)
56. Zhao, Q., Asif, M.S., Ma, Z.: Dnerv: Modeling inherent dynamics via difference neural representation for videos. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2031–2040 (2023)
57. Zou, N., Zhang, H., Cricri, F., Youvalari, R.G., Tavakoli, H.R., Lainema, J., Aksu, E., Hannuksela, M., Rahtu, E.: Adaptation and attention for neural video coding. In: 2021 IEEE International Symposium on Multimedia (ISM). pp. 240–244. IEEE (2021)

Supplementary material

A Trainable parameters

As shown in Tab. 4, our proposed methods use far fewer training parameters than the full fine-tuning method. This is mainly because our methods do not have parameters in the encoder and hyperprior parts that need training, allowing for a faster learning process. Additionally, our methods only need to update a relatively small set of parameters, increasing their overall efficiency. Specifically, the method that repeats parameters has a much smaller number of parameters compared to the other methods. This points to the high efficiency and practicality of our methods and strengthens their potential for effective use in various real-world situations.

B Adapter on hyperprior model

We conducted experiments on integrating our adapter structure with hyperprior models. As Fig. 8 shows, this did not lead to improvements in PSNR and MS-SSIM values. Furthermore, the compression performance got worse at lower bitrates. This can be attributed to the fact that the added structure makes more bits need to be transferred.

C GoP size variation

As demonstrated in prior researches [32,33], the commonly used practical Group of Pictures (GoP) size is close to 32. Thus, We also evaluated the performance with the GoP size set to 32, using the same Rate-distortion (RD) curve. The datasets for this evaluation remained the same, including UVG [42], MCL-JCV [49], and HEVC class B and C [4]. As shown in Fig. 9, the RD curves of PSNR and MS-SSIM displayed largely similar performance to the previous result with a smaller GoP size. Although there was a slight decrease in performance at lower bitrates, the overall performance remained consistent, demonstrating the robustness and applicability of our proposed methods to larger GoP sizes. This suggests that our proposed methods can be effectively applied even when the GoP size is increased, further enhancing the versatility of our method.

D Qualitative results

As indicated in Section 4.3, some datasets, especially those with cartoon-style or complex movements, pose challenges in reconstructing images. Therefore, we present the qualitative results for each dataset in Fig. 10, Fig. 11, and Fig. 12. The comparison is made at similar bpp settings, revealing that SSF [2] exhibits motion blur in complex domains. Moreover, full fine-tuning results in distortion

	Total params. (M)	Train params. (M)			
		Encoder	Hyperprior	Decoder	all
Full fine-tuning	34.24	12.70	16.59	4.95	34.24
Ours	35.03	0	0	0.79	0.79
Ours(repeat)	34.27	0	0	0.03	0.03

Table 4: Number of training parameters for video sequence instance-adaptation.

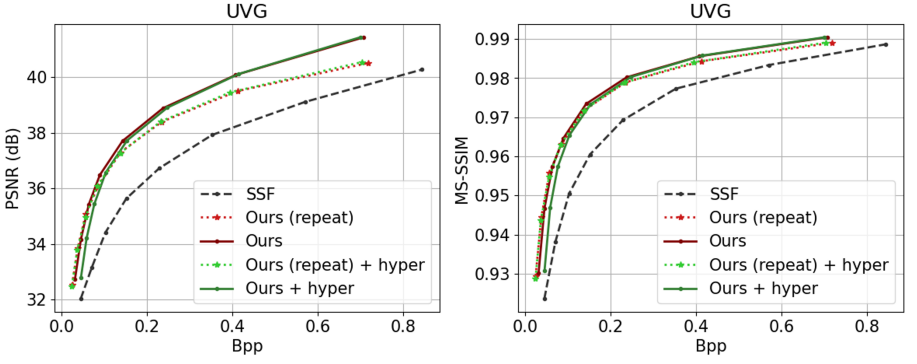


Fig. 8: RD-curve when apply adapter on hyperprior model. Comparison conducted on UVG dataset.

from the original, failing to accurately represent finer details. In contrast, both of our methods can effectively represent their respective areas without motion blur, even in the cartoon domain. These qualitative results highlight the superior overfitting mitigation capability of our methods.

E PSNR per frame

To assess the detailed performance of our method, we measure the PSNRs for each frame. As depicted in Fig. 13, the comparison is made between the baseline and our method with no duplication, focusing on the some of UVG dataset sequence. Both the baseline and our method show an increasing trend in PSNR. However, our method exhibits an overall improvement in PSNRs of approximately 1 dB, with smaller spikes, even though bpp is lower than baseline. This suggests that our method may be prone to overfitting the input video sequences with saving the number of bits.

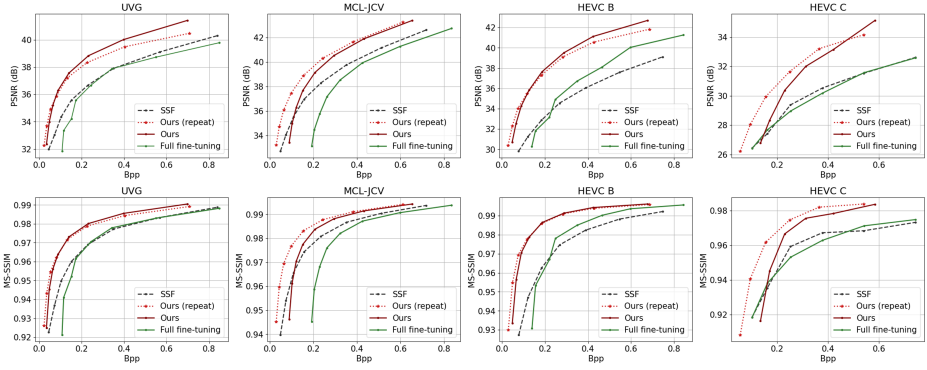


Fig. 9: RD-curve with GoP set to 32. Comparison conducted on UVG, MCL-JCV, HEVC class B, and C datasets.

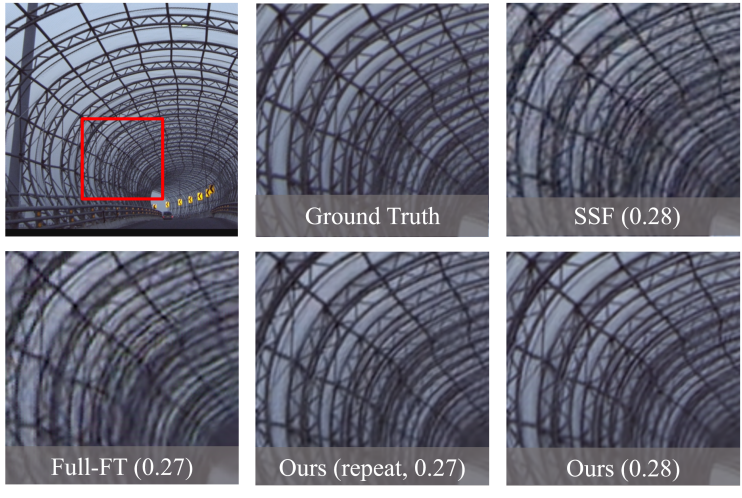


Fig. 10: Qualititive results of MCL-JCV 10 dataset.

Fig. 11: Qualititive results of MCL-JCV 24 dataset.

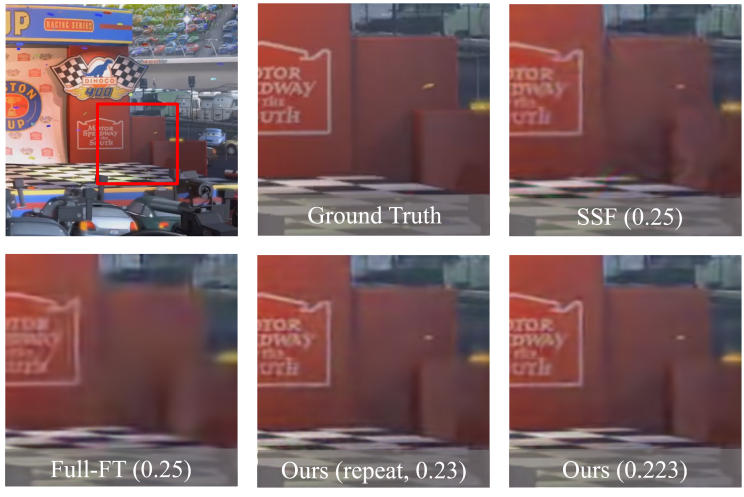


Fig. 12: Qualititive results of MCL-JCV 25 dataset.

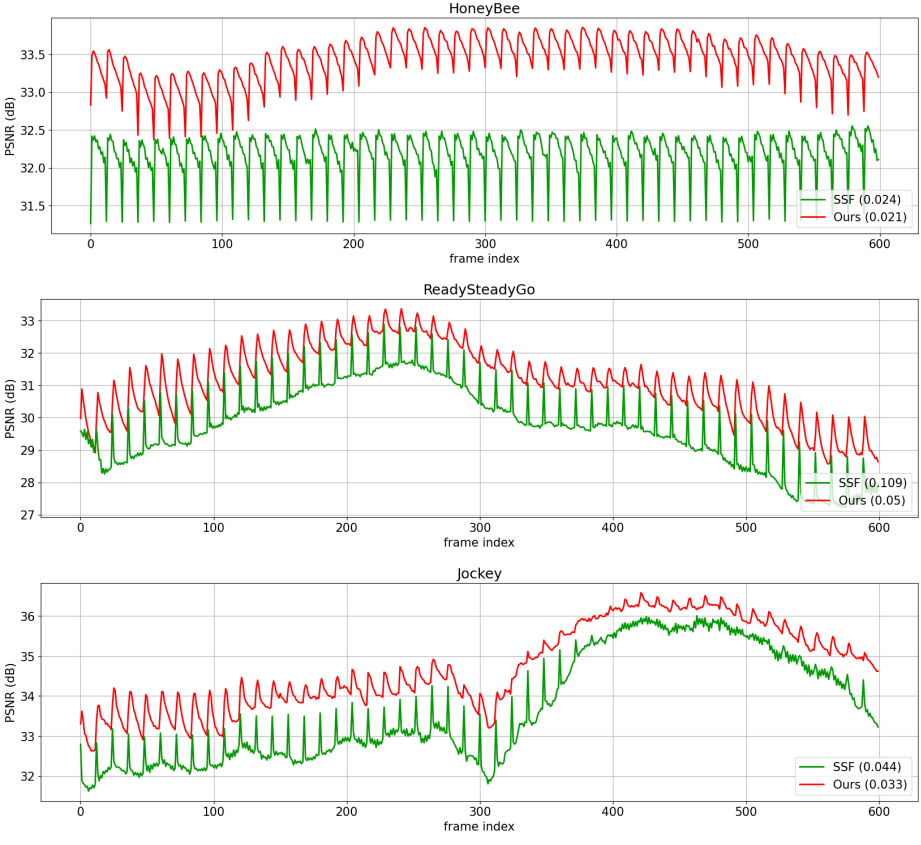


Fig. 13: PSNR for each frame using the same baseline model, tested on the 'HoneyBee', 'ReadySteadyGo', and 'Jockey' sequence. The Number in the legend represent bpp.