Complexity-guided Slimmable Decoder for Efficient Deep Video Compression

Zhihao Hu
Beihang University, China
huzhihao@buaa.edu.cn

Dong Xu ✉
University of Hong Kong, China
dongxu@hku.hk

Abstract

In this work, we propose the complexity-guided slimmable decoder (cgSlimDecoder) in combination with skip-adaptive entropy coding (SaEC) for efficient deep video compression. Specifically, given the target complexity constraints, in our cgSlimDecoder, we introduce a set of new channel width selection modules to automatically decide the optimal channel width of each slimmable convolution layer. By optimizing the complexity-rate-distortion related objective function to directly learn the parameters of the newly introduced channel width selection modules and other modules in the decoder, our cgSlimDecoder can automatically allocate the optimal numbers of parameters for different types of modules (e.g., motion/residual decoder and the motion compensation network) and simultaneously support multiple complexity levels by using a single learnt decoder instead of multiple decoders. In addition, our proposed SaEC can further accelerate the entropy decoding procedure in both motion and residual decoders by simply skipping the entropy coding process for the elements in the encoded feature maps that are already well-predicted by the hyperprior network. As demonstrated in our comprehensive experiments, our newly proposed methods cgSlimDecoder and SaEC are general and can be readily incorporated into three widely used deep video codecs (i.e., DVC, FVC and DCVC) to significantly improve their coding efficiency with negligible performance drop.

1. Introduction

Recently, learning-based video codecs [1, 15, 22, 28] have achieved promising coding performance and outperformed widely used commercial codecs like H.264 [46] and H.265 [40]. However, learning-based video codecs are always inefficient due to computationally complex operations. In practical application scenarios, it is desirable that the video codecs can decode the videos in real-time. Additionally, the decoders from different devices can afford different computational complexities under different scenarios. For example, a cloud server can afford higher computational resource while a smartphone can afford much less computational resource. Therefore, it is also desirable to develop a video decoder that can simultaneously support multiple computational complexities.

In order to achieve practical video codecs, two aspects should be taken into consideration during decoding. First, there are different modules in most decoders (see Figure 1 (a)) and the network structure of different modules in each decoder (*e.g.*, motion decoder, residual decoder and motion compensation networks) needs to be carefully designed to meet different complexity constraints. However, different modules need to handle different types of information (*e.g.*, residual information versus motion information) and thus require different channel widths under different complexity constraints, which makes it hard to manually design the optimal channel widths for different convolution layers in different modules. Second, the encoded bit-streams need to be entropy decoded back into the entire feature map. However, most elements in the feature map are already precisely predicted by the hyperprior networks [33]. As a result, it is less efficient to entropy code the entire feature map as it will also degrade the decoding efficiency.

Considering these two aspects, in this work, we propose a complexity-guided slimmable decoder (cgSlimDecoder) in combination with skip-adaptive entropy coding (SaEC) for efficient deep video coding. Motivated by the slimmable neural networks [21,53] for various visual recognition tasks, for the decoder network, we propose a complexity-guided slimmable decoder by additionally introducing a set of new channel width selection modules to automatically decide the optimal channel width of each slimmable convolution layer under different complexity constraints. By learning the parameters from our complexity-guided channel width selection modules and other modules (*e.g.*, the slimmable convolution layers) through optimizing the complexity-rate-distortion based objective function, our slimmable decoder can automatically select the optimal channel widths for different modules such that the target complexity constraints can be optimally allocated to different modules (*e.g.*, the motion/residual decoder and the motion compensation network). When the computational resource is sufficient (*resp.*, limited), larger (*resp.*, smaller) channel width is preferred in the decoding process to reconstruct high-quality video sequences (*resp.*, for more efficient decoding). In addition, we also propose the SaEC method for more efficient entropy coding of the encoded motion/residual feature map from the motion/residual encoder. Specifically, we take the hyperprior information as the input to predict whether each element can be precisely predicted by the hyperprior networks. For each precisely predicted element, we directly use the mean value of its predicted distribution from the hyperprior network, which will save bits and also accelerate the decoding process by skipping the entropy decoding process for these elements. The rest elements are still entropy coded for more accurate reconstruction.

To demonstrate the effectiveness of our proposed methods, we evaluate our cgSlimDecoder in combination with SaEC based on different baseline frameworks including DVC [28], FVC [15] and DCVC [22]. The experimental results demonstrate that our proposed methods are general and can significantly improve the decoding efficiency with comparable rate-distortion performance.

Our contributions are summarized as follows:

- We propose a complexity-guided slimmable decoder for efficient video decoding, which can simultaneously support multiple complexity levels by simply using one learned decoder instead of multiple decoders.
- Our newly proposed cgSlimDecoder method can automatically allocate the optimal complexities (*i.e.*, the optimal channel widths for different convolution layers) for different types of modules (*e.g.*, motion/residual decoder) to meet the overall complexity constraints of the entire video decoder. In addition, we also propose the SaEC coding method for more efficient and effective entropy coding.
- Our proposed methods are general and can be incorporated into a set of widely used video compression methods (*i.e.*, DVC, FVC and DCVC) and improve their decoding efficiency with negligible performance drop.

2. Related Work

2.1. Image and Video Codecs

Conventional image and video codecs (*e.g.*, H.264 [46] and H.265 [40]) heavily rely on hand-crafted operations. Recently, a large amount of learning-based image codecs [2–4, 6, 8, 9, 13, 33–35, 41–43, 48, 49, 54] have been proposed, in which some recent works [35, 48, 49, 54] have outperformed the state-of-the-art conventional image codecs like the intra codec of VTM [39]. Most of the recent works [9, 13, 33, 34] focus on how to improve the hyperprior networks for more accurate feature distribution prediction. Meanwhile, some learning-based video codecs [1, 7, 10, 11, 15, 16, 23, 24, 26–31, 36–38, 47, 52] were also proposed. Lu *et al.* [28] proposed the first end-to-end optimized video codec by replacing all the hand-crafted modules with learning-based modules. Hu *et al.* [15] proposed a feature space video coding framework by performing all major operations in the feature space. In DCVC [22]. Li *et al.* proposed the contextual coder to replace the residual coder. More recently, Mentzer *et al.* [31] proposed the transformer-based video coding framework without explicitly encoding the motion feature.

2.2. Efficient Coding

For the conventional codecs, the simplified versions like x264 and x265 were proposed for efficient video coding. Recently, several works studied how to improve the efficiency of learning-based image and video codecs. Johnston *et al.* [18] proposed several approaches for improving the efficiency of image coding. Guo *et al.* [12] used a single network to learn complexity and bit-rate adaptive image codec. Different from [12, 18], our cgSlimDecoder aims at optimally allocating the given complexity budget to different types of modules (*e.g.*, motion decoder and residual decoder), which is not considered in these deep image compression methods like [12, 18]. Note different types of modules play different roles in deep video compression, and the performance drop (due to utilization of computationally less expensive layers) in some modules (*e.g.*, motion decoder) may also affect the subsequent modules (*e.g.*, residual decoder), so how to allocate the given complexity budget over different types of modules is a non-trivial task. Rippel *et al.* [36] and Le *et al.* [20] have proposed efficient video codecs by *manually* designing the network structure for both motion and residual decoding networks and these methods [20, 36] can only support one complexity level. In contrast to [20, 36], in our work, multiple complexity levels can be readily supported by using one decoder and our cgSlimDecoder also *automatically* learns the optimal channel widths for different types of modules.

2.3. Slimmable Networks

Slimmable neural networks [21, 53] were recently proposed for balancing the trade-off between accuracy and model complexity. However, previous slimmable neural networks are mainly used for one single network and only consider the complexity-accuracy trade-off. It is still a non-trivial task to develop a slimmable neural network to decide the optimal channel widths for different types of modules (*e.g.*, residual decoder versus motion decoder) in video codecs by optimizing the complexity-rate-distortion trade-off. Recently, the slimmable neural networks are also used in deep image compression [51] and deep video com-

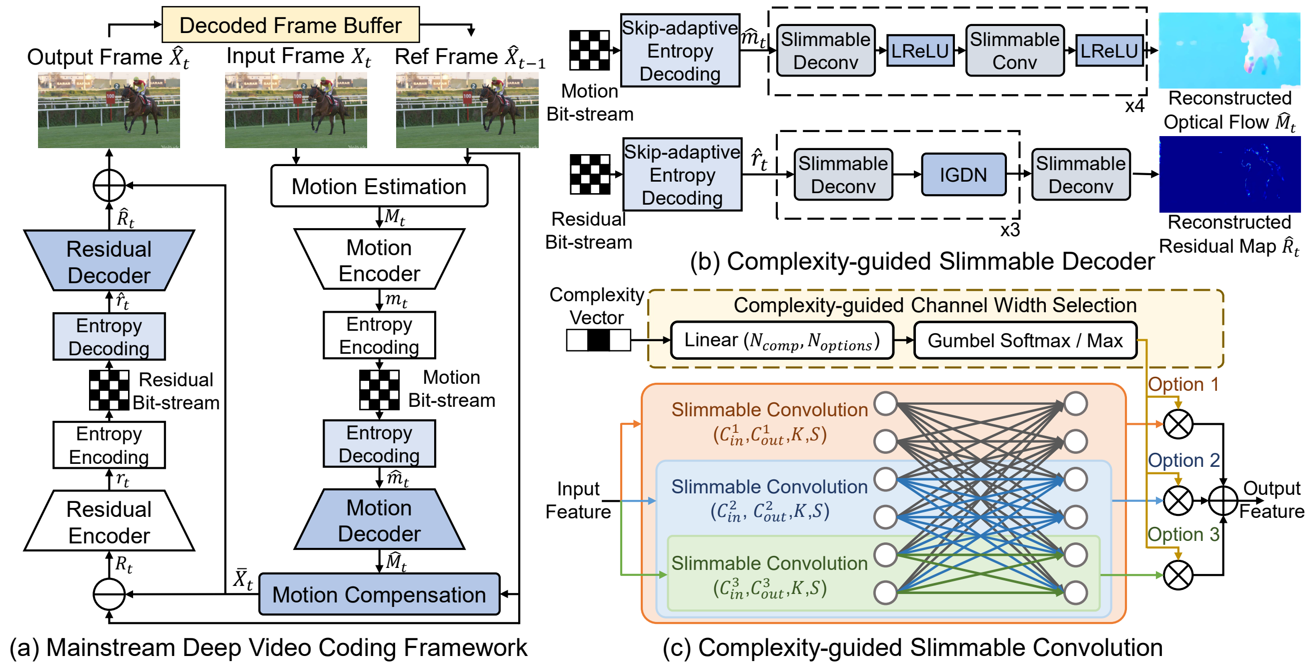

Figure 1. (a) Overview of the DVC [28] framework. Note other deep video compression frameworks follow the similar pipeline. Only the modules in blue are performed in the decoder side. In (b), taking DVC [28] as an example, we provide the detailed network structure of our proposed slimmable motion and residual decoder with skip-adaptive entropy decoding. (c) The details of our proposed complexity-guided slimmable convolution layer. Taking the complexity vector as the input, the channel width selection module will automatically select the optimal channel width for each slimmable convolution layer in order to optimally allocate the overall complexity budget to different types of modules. "Slimmable Convolution (C_{in},C_{out},K,S)" denotes the slimmable convolution layer with C_{in} input channels, C_{out} output channels, the kernel size K and the stride S. "LReLU" denotes the LeakyReLU operation.

pression [25] to achieve variable bit-rate coding, which is intrinsically different from our work as we aim to learn one decoder for efficient deep video compression. In addition, they *manually* design the channel width for each layer and their codec at low complexity level (*resp.*, high complexity level) can only be used for low bit-rate coding (*resp.*, high bit-rate coding). In contrast to [25, 51], at each given bit-rate, our proposed cgSlimDecoder can support multiple complexity levels by using one single decoder and our work can also *automatically* learn the optimal channel bandwidths from different modules.

3. Methodology

3.1. Overall Framework

Problem Formulation. Taking the video sequence $\{X_1, X_2, ..., X_t, ..., X_n\}$ as the input, the goal of the video codec is to reconstruct the high quality video sequence under different bit-rate constraints.

In this section, we take DVC [28] as an example to illustrate the overall coding procedure of the mainstream deep video coding framework, which is also shown in Figure 1 (a).

Motion Estimation. At the encoder side, taking the input frame X_t and the reference frame $\hat{X}_{t-1}$ as the input, the motion estimation network (*e.g.*, the optical flow estimation network) will estimate the motion information M_t.

Motion Compression. The motion compression module includes the motion encoder, entropy encoding, entropy decoding and motion decoder. The motion information M_t from the motion estimation module is encoded by the motion encoder to generate the encoded motion feature m_t. Then the entropy encoding operation is performed to convert the encoded motion feature m_t into motion bit-stream, which will be sent to the decoder side. At the decoder side, the motion bit-stream is firstly entropy decoded into the entropy decoded motion feature $\hat{m}_t$, which is then decoded into the decoded motion information $\hat{M}_t$ by using the motion decoder network.

Motion Compensation. Given the reference frame $\hat{X}_{t-1}$ and the decoded motion information $\hat{M}_t$, the motion compensation module is performed to generate the predicted frame $\bar{X}_t$.

Residual Compression. The residual information R_t is generated based on the input frame X_t and the predicted frame $\bar{X}_t$. The coding procedure of the residual information R_t is similar to the coding procedure of the motion information M_t, which includes the residual encoder, entropy encoding, entropy decoding and residual decoder.

Video Decoding. In Figure 1 (a), the modules marked in blue are used for video decoding. The decoder receives the motion bit-stream and the residual bit-stream of each frame. The entropy decoding process is first performed to generate the entropy decoded motion feature $\hat{m}_t$ and the entropy decoded residual feature $\hat{r}_t$ from the bit-streams. Then the motion decoder network and the residual decoder network decode the entropy decoded features $\hat{m}_t$ and $\hat{r}_t$ back to the decoded motion information $\hat{M}_t$ and the decoded residual information $\hat{R}_t$, respectively. Then the motion compensation module is performed to generate the predicted frame $\bar{X}_t$. Finally, the reconstructed output frame $\hat{X}_t$ is generated by adding the reconstructed residual information $\hat{R}_t$ back to the predicted frame $\bar{X}_t$.

The video decoder consists of multiple modules (*e.g.*, motion decoder, residual decoder and motion compensation networks) and each of them contains multiple convolution layers, which makes it hard to manually design the optimal channel width at each layer in different modules to satisfy different complexity constraints on various devices. Additionally, the previous works [15,22,28] always directly perform the entropy coding operation on the entire feature map without considering the efficiency issue. Therefore, it is desirable to develop a complexity-guided slimmable decoder with efficient entropy coding for video codecs.

3.2. Complexity-guided Slimmable Decoder

Recently, a large amount of deep video codecs [1, 15, 22, 28] are proposed for better video coding performance. However, these networks are designed without considering the complexity constraints on various devices. It is also impractical to design multiple decoder network structures for multiple complexity constraints. Therefore, we propose the complexity-guided slimmable decoder (cgSlimDecoder) to satisfy various complexity constraints by using one single decoder. In Figure 1 (b), we provide the detailed network structure of our proposed complexity-guided slimmable decoder. In our proposed decoder, each convolution/deconvolution layer is replaced by the slimmable convolution/deconvolution layer for achieving multiple complexity constraints.

The details of the complexity-guided slimmable convolution layers are provided in Figure 1 (c). The complexity vector is a one-hot vector to represent the current complexity constraint, which is taken as the input of the complexity-guided channel width selection module. Then the channel width selection module will decide the optimal channel width of each convolution layer under the current complexity constraint. In this work, three options are provided for each convolution layer for achieving different computational complexities. For example, for the first option, all the channels are selected with full complexity. When the first option is selected for all convolution layers, the network is the same as those networks without using the slimmable convolution layers. For the second or the third option, only a subset of the convolution weights is used in the convolution operation to achieve less computational complexity. In the slimmable convolution layer from Figure 1 (c), the second option uses $\frac{2}{3}$ input channels and only outputs $\frac{2}{3}$ output channels. Therefore, the complexity of this option is only $\frac{4}{9}$ of the original complexity. The third option in Figure 1 (c) only uses $\frac{1}{3}$ of the input and output channel, which achieves $\frac{1}{9}$ complexity of the original computational complexity. Note that when option 2 or 3 is selected, the channel number of the output feature map is less than the original output channel number. To address this issue, we directly set each element of the output feature map as the element of the input feature map at the corresponding channel instead of setting them to zeros in order to prevent the loss of information.

Complexity-guided Channel Width Selection. Taking the complexity vector as the input, our complexity-guided channel width selection module will separately decide the channel width for each convolution layer in the video decoder. However, directly selecting the channel width option with the maximum probability is undifferentiable, which makes the optimization of the channel width selection module infeasible. Therefore, we adopt the Gumbel Softmax strategy [17] to decide the optimal width of each convolution layer during the training procedure. During the testing process, we directly select the channel width option with the maximum probability.

3.3. Skip-adaptive Entropy Coding

For practical learning-based video coding, the efficiency of entropy coding should also be considered. Inspired by [38], we propose skip-adaptive entropy coding (SaEC) by automatically predicting the skip mode for each element of the encoded motion feature map.

In Figure 2 (a), we take the encoded motion feature m_t as an example to illustrate the procedure of our proposed skip-adaptive entropy coding method. We assume the size of the encoded motion feature m_t is $c \times h \times w$, which denotes that the feature m_t has "c" channels with the spatial size of "$h \times w$". Therefore, the hyperprior network will predict the hyperprior information (*i.e.*, the mean and the variance) for each element of the encoded motion feature m_t, which has the same spatial size $h \times w$ with $2 \times c$ channels. We take the hyperprior information as the input of the mode prediction network. As shown in Figure 2 (b), similar to the channel width selection module, we also use the Gumbel Softmax strategy [17] to solve the gradient undifferentiable issue during training and then select the skip mode for each element of the encoded feature map m_t. The mode prediction module will generate a 0/1 mask for each element of the feature map m_t. "1" in the predicted 0/1 mask denotes the corresponding element is uncertain and should be entropy

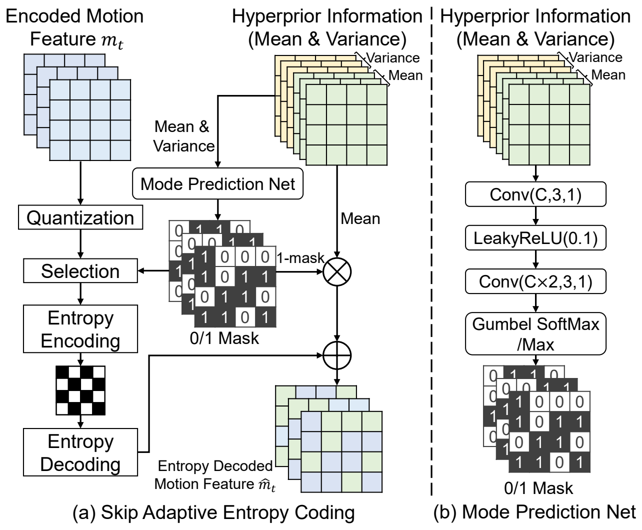

Figure 2. (a) Details of our proposed skip-adaptive entropy coding method. Given the encoded motion feature m_t and its corresponding hyperprior information (*i.e.* the mean and variance for each element in m_t), we automatically select the coding mode (*i.e.*, the skip mode) for each element of the encoded motion feature m_t. (b) The detailed network structure of the mode prediction network. We use the Gumbel Softmax strategy during training and use the "Max" operation during testing. "Conv(C,K,S)" denotes the convolution layer with "C" output channels, the kernel size "K" and the stride "S".

coded. "0" denotes that the symbol is well predicted by the hyperprior network so we can directly use the mean value in order to accelerate the entropy coding procedure and save bits. Our proposed SaEC method in residual compression follows the same procedure as that in motion compression.

Therefore, by using our proposed SaEC module, the quantized motion and residual features can be efficiently and effectively entropy decoded for practical deep video coding. The previous work C2F [14] proposed a hyperprior-guided mode prediction network to predict the skip mode for effective residual coding, which set the skipped elements as zeros. Different from C2F [14], we use the skip-adaptive entropy coding for both motion and residual entropy coding and set the skipped elements as the mean values from the hyperprior network. Additionally, we also use a much simpler and more efficient mode prediction network.

3.4. Two-stage Training Strategy for Complexity-guided Slimmable Decoder (cgSlimDecoder)

Although our newly proposed complexity-guided slimmable decoder can automatically select the optimal width for each convolution layer, it is a non-trivial task to optimize the entire video coding network to achieve multiple complexity constraints without significantly degrading the rate-distortion performance. Therefore, we propose a two-stage optimization strategy to optimize the video compression method with our cgSlimDecoder.

At the first stage, we randomly initialize the weight/parameter of each complexity-guided channel width selection module and train the whole network without complexity penalty. The loss function at this stage is formulated as follows,

$$L = R_m + R_r + \lambda D(\hat{X}_t, X_t) \tag{1}$$

in which the R_m and R_r denote the bit-rate cost for entropy coding the motion and residual information, respectively. $D(\hat{X}_t, X_t)$ denotes the distortion between the output frame $\hat{X}_t$ and the input frame X_t. λ is the hyper-parameter to control the trade-off between the bit-rate cost and the distortion. At this stage, as the channel width selection module is randomly initialized without complexity penalty, the channel width selection modules will randomly select the channel width for each layer such that the video coding networks with different channel width options are trained.

For the second stage, we add the complexity penalty for achieving different complexity constraints. N complexity targets are predefined as $C_1^{tar}, C_2^{tar}, ..., C_N^{tar}$. For each training step, we randomly select a complexity target C_i^{tar} and feed the corresponding one-hot complexity vector to the network. For the i^{th} complexity target C_i^{tar}, the optimization function is formulated as follows,

$$\theta^* = \arg\min_{\theta} R_m + R_r + \lambda D(\hat{X}_t, X_t), s.t. C_i \leq C_i^{tar} \tag{2}$$

in which R_m, R_r, λ and $D(\hat{X}_t, X_t)$ are the same as in Eq. (1). θ is the network parameters, θ^* is our target network parameters and C_i is the decoder complexity at the current step. We solve the optimization problem in Eq. (2) by using the Lagrangian multiplier based optimization technology. Therefore, the optimization problem in Eq. (2) can be rewritten as follows,

$$\theta^* = \arg\min_{\theta} R_m + R_r + \lambda D(\hat{X}_t, X_t) + \alpha_i C_i \tag{3}$$

where α_i is the Lagrangian multiplier. Therefore, the loss function of the second stage is formulated as follows,

$$L = R_m + R_r + \lambda D(\hat{X}_t, X_t) + \alpha_i C_i \tag{4}$$

The details on how to set the weight α_i will be discussed in the supplementary materials.

3.5. Implementation Details

In this work, we incorporate the newly proposed cgSlimDecoder in combination with the SaEC method into different baseline frameworks including DVC [28], FVC [15] and DCVC [22]. For the residual blocks in these three frameworks, we only use one channel width selection module to select the channel width for all convolution layers of one residual block for more stable training. Considering that the hyperprior networks will only bring negligible computational complexity cost, we do not use slimmable convolution layers in the hyperprior decoder. The details of different baseline frameworks are also provided.

For the DVC [28] framework, in Figure 1 (b), we provide the detailed network design for the motion decoder and the residual decoder. The decoder of DVC [28] also includes the heavy motion compensation network. For the motion compensation network, we replace all the convolution layers with slimmable convolution layers. Considering that DVC does not have the hyperprior network to predict the mean and variance for the encoded motion and residual features, we directly take the encoded motion/residual feature as the input and predict the coding mode for each 4×4 block, which will be transmitted to the decoder.

For the FVC [15] framework, we remove the time-consuming multi-frame feature fusion module. All the convolution layers in the feature encoder, feature decoder, offset decoder, motion compensation and residual decoder are replaced by using the slimmable convolution layers.

For the DCVC [22] framework, we remove the time-consuming auto-regressive context model. All the convolution layers in the motion decoder, the motion refinement, the feature extraction, the context refinement, the context encoder and the contextual decoder modules are replaced by using the slimmable convolution layers.

4. Experiments

4.1. Experimental Setup

Training Dataset. We use the Vimeo-90K dataset as the training [50] set, which consists of 89,800 video sequences. Each video sequence contains seven frames with the resolution of 448×256. We randomly flip and crop the video sequence to the resolution of 256×256.

Testing Datasets. We evaluate our methods on the HEVC Class B, C, D, E datasets [40], the UVG dataset [32] and the MCL-JCV dataset [44]. The HEVC Class B, C, D, E datasets consist of video sequences with the resolutions of 1920×1080, 832×480, 416×240 and 1280×720, respectively, which are frequently used for the evaluation of conventional video codecs. The UVG dataset [32] consists of seven high frame-rate video sequences with the resolution of 1920×1080. The MCL-JCV dataset [44] is also widely used for evaluating the coding performance, which contains thirty 1080p video sequences with various contents. Due to space limitation, the results on the HEVC datasets will be provided in the supplementary materials.

Evaluation Metric. PSNR and MS-SSIM [45] are used for evaluating the quality of the reconstructed video sequence. PSNR is the most widely used metric for evaluating distortion. MS-SSIM is a more subjective visual quality evaluation metric. We use the bit per pixel (bpp) to represent the bit-rate cost. The MACs (multiply–accumulate operations) are used to evaluate the computational complexity of different models. Considering that we need three metrics (*i.e.*, MACs, bpp, PSNR/MS-SSIM) to evaluate our method, we use the BD-PSNR [5] and BD-MSSSIM (dB) to evaluate the average PSNR/MS-SSIM(dB) improvement over two rate-distortion curves.

Training Details. We use the same training procedure for all the baseline methods including DVC [28], FVC [15] and DCVC [22]. The hyper-parameter λ is set as $2048, 1024, 512$ and 256. For training the video compression model with the highest λ value, we use 2,000,000 steps to train the video compression model with two consecutive frames. After that, we randomly sample 5 frames from the training video sequence and train the model with 200,000 steps. Finally, we fine-tune the models with other λ values for 50,000 steps. We set the initial learning rate for DVC, FVC and DCVC as 1e-4, 5e-5 and 1e-4, respectively. Then the learning rate is decreased by 80% at 1,800,000 and 2,100,000 steps. The same training strategy is used for the baseline methods equipped with our proposed methods and we fix the channel width selection module when fine-tuning the models with lower λ values. As described in Section 3.4, we additionally indicate the complexity penalty after the first 500,000 training steps. The initial τ in each Gumbel Softmax layer is set as 4 and is gradually reduced to zero. We set the batch size as four and use the Adam [19] optimizer for training. Our code is implemented based on Pytorch with CUDA support.

We use the MSE loss as the distortion loss in our loss functions. For our proposed complexity-guided slimmable decoder, we use three target complexity levels. For each slimmable convolution/deconvolution layer, we use three channel width options including 100%, 50% and 25% of the original channel width.

4.2. Complexity-Rate-Distortion Performance

Baseline Methods. We use three widely used deep video codecs including DVC [28], FVC [15] and DCVC [22] as the baseline methods. We name DVC / FVC / DCVC in combination with our cgSlimDecoder (*resp.*, cgSlimDecoder+SaEC) as cgSlimbDecoder(DVC) / cgSlimbDecoder(FVC) / cgSlimbDecoder(DCVC) (*resp.*, cgSlimbDecoder+SaEC(DVC) / cgSlimbDecoder+SaEC(FVC) / cgSlimbDecoder+SaEC(DCVC)). To compare with our newly proposed cgSlimDecoder in combination with SaEC, we use a simple baseline method by directly reducing a fixed ratio (*i.e.*, 25% and 50%) of the original channel width for each convolution layer in the decoder network. For fair comparison, the channel numbers of the convolutions in the hyperprior network are not reduced. The baseline methods are also well-trained by fine-tuning the pretrained DVC / FVC / DCVC modules when only using 75% and 50% of the original channel width. Considering that it is hard to di-

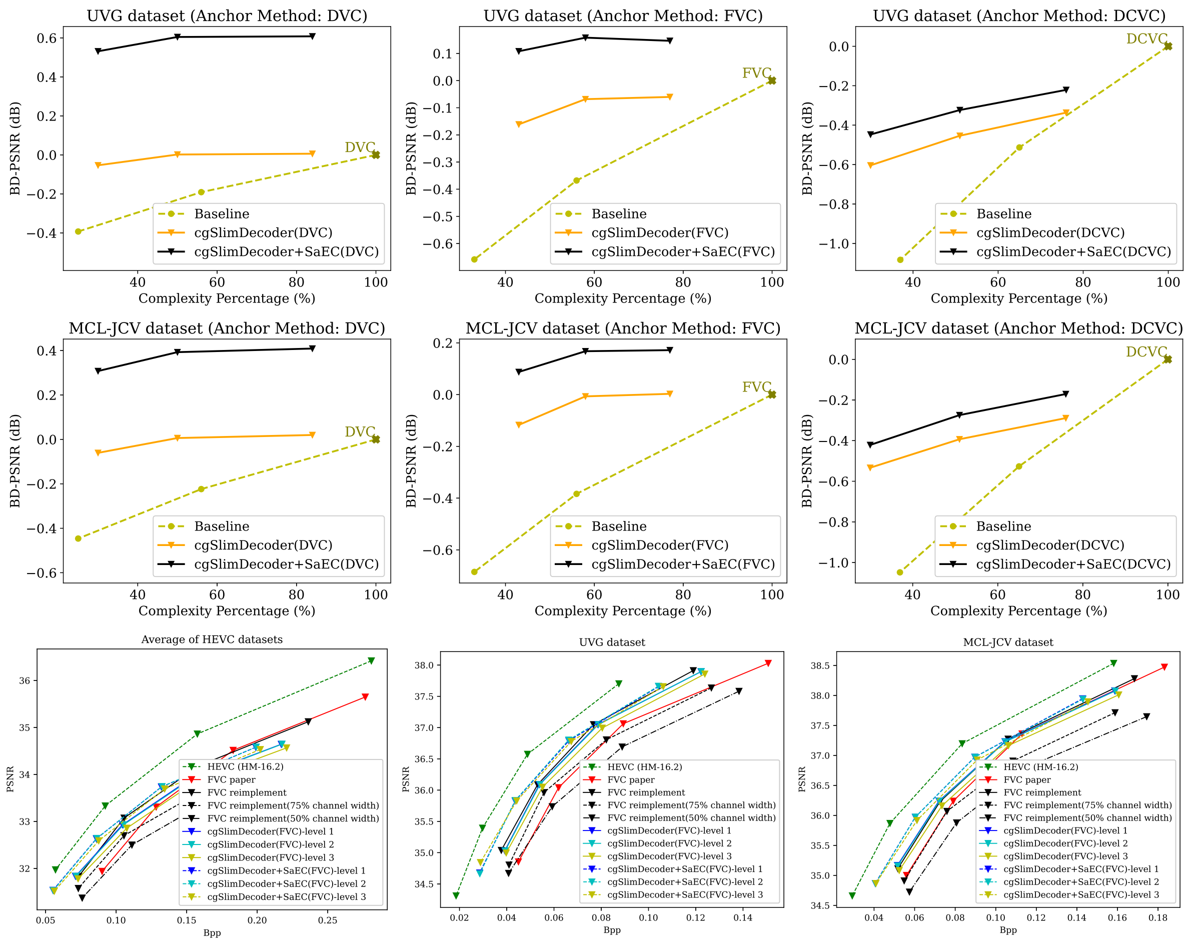

Figure 3. The experimental results of different methods on the HEVC, UVG and MCL-JCV datasets.

rectly compare the complexity-rate-distortion performance, we use the corresponding methods DVC, FVC and DCVC with all original channel widths as the anchor method and calculate the BD-PSNR [5] results under different complexity constraints to make a fair comparison. BD-PSNR denotes the average PSNR improvement when compared with the anchor method. Therefore, higher BD-PSNR result denotes better rate-distortion performance.

As shown in Figure 3, our cgSlimDecoder(DVC) and cgSlimDecoder(FVC) achieve comparable BD-PSNR results when compared with the anchor method DVC and FVC. For example, cgSlimDecoder(DVC) drops less than 0.1dB at the lowest complexity level on both UVG and MCL-JCV datasets. We observe that the simple baseline method by directly reducing a fixed ratio of channel widths brings much more performance drop than our proposed cgSlimDecoder. For example, the performance of the baseline method FVC at the middle complexity level drops about 0.4dB on the MCL-JCV dataset, while our proposed cgSlimDecoder(FVC) only drops less than 0.1dB at the lowest complexity level when compared with the anchor method FVC with the original channel widths. DCVC is a more carefully designed method, which leads to much more performance drop when less complexity is used in the decoder. However, our method still achieves much better performance than the baseline method DCVC. When our cgSlimDecoder(DCVC) uses about 50% of the total complexity on the MCL-JCV dataset, the performance drops about 0.4dB, while the baseline method with the middle complexity level drops about 0.6dB. Additionally, our cgSlimDecoder in combination with skip-adaptive entropy coding (SaEC) can further improve the coding performance with faster entropy decoding speed. On the UVG dataset, our cgSlimDecoder+SaEC(DCVC) achieves 0.2dB performance improvement when compared with our cgSlimDecoder(DCVC).

Rate-Distortion Curves. Considering providing all the RD curves in one plot makes it hard to compare the performance, we take FVC as an example and provide the rate-distortion curves in the last row of Fig. 3. Note FVC (reimplement) removes the time-consuming multi-frame feature fusion and achieves better results than FVC (paper).

Table 1. The complexity percentages of different modules in the original DCVC [22] method and our cgSlimDecoder(DCVC) at different complexity levels based on different input complexity vectors.

Complexity Levels	Original	Level 1	Level 2	Level 3
Motion Decoder	7%	1%	2%	3%
Motion Refinement	25%	27%	34%	18%
Feature Extraction	10%	13%	15%	25%
Context Refinement	15%	5%	8%	13%
Context Encoder	5%	3%	4%	4%
Contextual Decoder	39%	51%	38%	37%

Table 2. Skip percentage in both motion and residual entropy coding of our proposed cgSlimDecoder+SaEC(FVC) on the MCL-JCV dataset at different λ values.

	λ=2048	λ=1024	λ=512	λ=256
Motion	99.60%	99.74%	99.79%	99.83%
Residual	85.90%	91.54%	94.27%	96.21%

4.3. Model Analysis

Complexity Percentage of Different Modules. In Table 1, we take the DCVC [22] as an example to provide the complexity percentages of different modules in the original DCVC and our proposed cgSlimDecoder(DCVC) based on different input complexity levels. It is observed that at the highest complexity level (*i.e.*, level 1), our method cgSlimDecoder(DCVC) allocates more than 50% complexity to the contextual decoder. While the motion decoder only takes less than 5% percentage of the total complexity at all complexity levels. One possible explanation is that contextual decoding is more important than motion decoding, namely, reducing the convolution channel widths in the motion decoder will have less influence than reducing the convolution channel widths in the contextual decoder, which demonstrates that our proposed complexity-guided slimmable decoder can automatically allocate the optimal complexity to different modules in order to meet different complexity constraints.

Skip Percentage for Motion and Residual Entropy Coding. In Table 2, we also provide the skip percentage of our proposed skip-adaptive entropy coding (SaEC) at different λ values on the MCL-JCV dataset. We take cgSlimDecoder+SaEC(FVC) as an example, in which we set the highest complexity level for our cgSlimDecoder(FVC). We observe that more than 99.5% elements are skipped in motion entropy coding and more than 85% elements are skipped in residual entropy coding, which can significantly improve the entropy coding speed. For smaller λ values, lower bitrates are required and thus the motion and residual features contain less information, which leads to larger skip percentages for both motion and residual entropy coding. For example, when setting $\lambda = 256$, 99.8% elements are skipped during motion entropy coding and 96.2% elements are skipped during residual entropy coding. The results demonstrate that our proposed SaEC can skip a large percentage of elements to avoid motion and residual entropy coding for these elements, which can also significantly improve the decoding efficiency.

Table 3. Decoding time of DVC, FVC and DCVC and our proposed cgSlimDecoder(DVC), cgSlimDecoder(FVC) and cgSlimDecoder(DCVC) when using the 1080p videos as input. “Level 1”, “Level 2” and “Level 3” indicate the complexity at the highest, the middle and the lowest level, respectively.

	Original	Level 1	Level 2	Level 3
DVC	163ms	140ms	107ms	79ms
FVC	127ms	107ms	89ms	71ms
DCVC	283ms	237ms	189ms	144ms

Running Time. The decoding times of our methods are provided in Table 3. We observe that our proposed cgSlimDecoder can significantly reduce the decoding time. For example, our cgSlimDecoder(DCVC)-level 3 reduces about 50% of the decoding time, which demonstrates the effectiveness of our cgSlimDecoder.

Running Time Reduction of our SaEC. To demonstrate the effectiveness of our proposed SaEC, we use the entropy coder “torchac” for evaluating the speed of entropy coding. Taking DCVC as an example, when SaEC is not used in our cgSlimDecoder(DCVC), the entropy decoding times of motion/residual coding are 147ms/33ms per frame on the MCL-JCV dataset, while the corresponding time is only 0.6ms/2ms when SaEC is adopted as most of the elements as skipped. It is observed that our proposed SaEC can reduce a large percentage of entropy coding time.

5. Conclusion

In this work, we have proposed a complexity-guided slimmable decoder (cgSlimDecoder) in combination with skip-adaptive entropy coding (SaEC) for efficient video coding. To support multiple complexity constraints in a single decoder, our proposed cgSlimDecoder can automatically select the optimal channel width for different convolution layers in different modules (*e.g.*, motion/residual decoder, motion compensation network). For efficient entropy coding, our SaEC can automatically skip the entropy coding procedures for a set of selected elements that are already well-predicted by the hyperprior network and directly use the predicted mean values from the hyperprior network for these elements. The experiments demonstrate that our proposed cgSlimDecoder in combination with SaEC is general and can be applied to three widely used video codecs including DVC, FVC and DCVC for achieving efficient video decoding under multiple complexity constraints with negligible rate-distortion performance drop.

Acknowledgement This work was supported by The Hong Kong Jockey Club Charities Trust under Grant 2022-0174 and in part by the Startup Funding from The University of Hong Kong.

References

[1] Eirikur Agustsson, David Minnen, Nick Johnston, Johannes Ballé, Sung Jin Hwang, and George Toderici. Scale-space flow for end-to-end optimized video compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8503–8512, 2020. 1, 2, 4

[2] Mohammad Haris Baig, Vladlen Koltun, and Lorenzo Torresani. Learning to inpaint for image compression. In *Advances in Neural Information Processing Systems*, pages 1246–1255, 2017. 2

[3] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *International Conference on Learning Representations (ICLR)*, 2017. 2

[4] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *International Conference on Learning Representations (ICLR)*, 2018. 2

[5] Gisle Bjontegaard. Calculation of average psnr differences between rd-curves. *VCEG-M33*, 2001. 6, 7

[6] Zhenghao Chen, Shuhang Gu, Guo Lu, and Dong Xu. Exploiting intra-slice and inter-slice redundancy for learning-based lossless volumetric image compression. *IEEE Transactions on Image Processing*, 2022. 2

[7] Zhenghao Chen, Guo Lu, Zhihao Hu, Shan Liu, Wei Jiang, and Dong Xu. LSVC: A learning-based stereo video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 734–743, 2022. 2

[8] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learning image and video compression through spatial-temporal energy compaction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10071–10080, 2019. 2

[9] 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*, pages 7939–7948, 2020. 2

[10] Abdelaziz Djelouah, Joaquim Campos, Simone Schaub-Meyer, and Christopher Schroers. Neural inter-frame compression for video coding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6421–6429, 2019. 2

[11] Runsen Feng, Yaojun Wu, Zongyu Guo, Zhizheng Zhang, and Zhibo Chen. Learned video compression with feature-level residuals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 120–121, 2020. 2

[12] Jinyang Guo, Dong Xu, and Guo Lu. Cbanet: Towards complexity and bitrate adaptive deep image compression using a single network. *arXiv preprint arXiv:2105.12386*, 2021. 2

[13] 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*, pages 14771–14780, 2021. 2

[14] Zhihao Hu, Guo Lu, Jinyang Guo, Shan Liu, Wei Jiang, and Dong Xu. Coarse-to-fine deep video coding with hyperprior-guided mode prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1704–1713, 2022. 5

[15] 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*, pages 1502–1511, 2021. 1, 2, 4, 5, 6

[16] Zhihao Hu, Dong Xu, Guo Lu, Wei Jiang, Wei Wang, and Shan Liu. Fvc: An end-to-end framework towards deep video compression in feature space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 2

[17] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *International Conference on Learning Representations (ICLR)*, 2017. 4

[18] Nick Johnston, Elad Eban, Ariel Gordon, and Johannes Ballé. Computationally efficient neural image compression. *arXiv preprint arXiv:1912.08771*, 2019. 2

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference for Learning Representations*, 2015. 6

[20] Hoang Le, Liang Zhang, Amir Said, Guillaume Sautiere, Yang Yang, Pranav Shrestha, Fei Yin, Reza Pourreza, and Auke Wiggers. Mobilecodec: neural inter-frame video compression on mobile devices. In *Proceedings of the 13th ACM Multimedia Systems Conference*, pages 324–330, 2022. 2

[21] Changlin Li, Guangrun Wang, Bing Wang, Xiaodan Liang, Zhihui Li, and Xiaojun Chang. Dynamic slimmable network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8607–8617, 2021. 1, 2

[22] Jiahao Li, Bin Li, and Yan Lu. Deep contextual video compression. *Advances in Neural Information Processing Systems*, 34, 2021. 1, 2, 4, 5, 6, 8

[23] Jiahao Li, Bin Li, and Yan Lu. Hybrid spatial-temporal entropy modelling for neural video compression. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 1503–1511, 2022. 2

[24] Jianping Lin, Dong Liu, Houqiang Li, and Feng Wu. M-LVC: Multiple frames prediction for learned video compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3546–3554, 2020. 2

[25] Zhaocheng Liu, Luis Herranz, Fei Yang, Saiping Zhang, Shuai Wan, Marta Mrak, and Marc Górriz Blanch. Slimmable video codec. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1743–1747, 2022. 3

[26] Salvator Lombardo, Jun Han, Christopher Schroers, and Stephan Mandt. Deep generative video compression. In *Advances in Neural Information Processing Systems*, pages 9287–9298, 2019. 2

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

[28] 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 Conference on Computer Vision and Pattern Recognition*, pages 11006–11015, 2019. 1, 2, 3, 4, 5, 6

[29] 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*, in Press:1–1, 2020. 2

[30] Fabian Mentzer, Eirikur Agustsson, Johannes Ballé, David Minnen, Nick Johnston, and George Toderici. Towards generative video compression. *arXiv preprint arXiv:2107.12038*, 2021. 2

[31] Fabian Mentzer, George Toderici, David Minnen, Sung-Jin Hwang, Sergi Caelles, Mario Lucic, and Eirikur Agustsson. Vct: A video compression transformer. *arXiv preprint arXiv:2206.07307*, 2022. 2

[32] A. Mercat, Marko Viitanen, and J. Vanne. UVG dataset: 50/120fps 4k sequences for video codec analysis and development. *Proceedings of the 11th ACM Multimedia Systems Conference*, 2020. 6

[33] David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, pages 10771–10780, 2018. 1, 2

[34] David Minnen and Saurabh Singh. Channel-wise autoregressive entropy models for learned image compression. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 3339–3343. IEEE, 2020. 2

[35] Guanbo Pan, Guo Lu, Zhihao Hu, and Dong Xu. Content adaptive latents and decoder for neural image compression. 2022. 2

[36] Oren Rippel, Alexander G Anderson, Kedar Tatwawadi, Sanjay Nair, Craig Lytle, and Lubomir Bourdev. ELF-VC: Efficient learned flexible-rate video coding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3033–3042, 2021. 2

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

[38] Yibo Shi, Yunying Ge, Jing Wang, and Jue Mao. Alphavc: High-performance and efficient learned video compression. *arXiv preprint arXiv:2207.14678*, 2022. 2, 4

[39] Gary Sullivan. Versatile video coding (VVC) arrives. In *2020 IEEE International Conference on Visual Communications and Image Processing (VCIP)*, pages 1–1. IEEE, 2020. 2

[40] 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. 1, 2, 6

[41] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. *International Conference for Learning Representations*, 2017. 2

[42] George Toderici, Sean M O'Malley, Sung Jin Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell, and Rahul Sukthankar. Variable rate image compression with recurrent neural networks. *International Conference for Learning Representations*, 2017. 2

[43] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. Full resolution image compression with recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5306–5314, 2017. 2

[44] Haiqiang Wang, Weihao Gan, Sudeng Hu, Joe Yuchieh Lin, Lina Jin, Longguang Song, Ping Wang, Ioannis Katsavounidis, Anne Aaron, and C-C Jay Kuo. MCL-JCV: a jnd-based H.264/AVC video quality assessment dataset. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 1509–1513. IEEE, 2016. 6

[45] 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, pages 1398–1402. Ieee, 2003. 6

[46] 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. 1, 2

[47] Chao-Yuan Wu, Nayan Singhal, and Philipp Krahenbuhl. Video compression through image interpolation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 416–431, 2018. 2

[48] Yaojun Wu, Xin Li, Zhizheng Zhang, Xin Jin, and Zhibo Chen. Learned block-based hybrid image compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 2021. 2

[49] Yueqi Xie, Ka Leong Cheng, and Qifeng Chen. Enhanced invertible encoding for learned image compression. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 162–170, 2021. 2

[50] 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. 6

[51] Fei Yang, Luis Herranz, Yongmei Cheng, and Mikhail G Mozerov. Slimmable compressive autoencoders for practical neural image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4998–5007, 2021. 2, 3

[52] Ren Yang, Fabian Mentzer, Luc Van Gool, and Radu Timofte. Learning for video compression with hierarchical quality and recurrent enhancement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6628–6637, 2020. 2

[53] Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1803–1811, 2019. 1, 2

[54] Yinhao Zhu, Yang Yang, and Taco Cohen. Transformer-based transform coding. In *International Conference on Learning Representations*, 2021. 2