Towards Practical Real-Time Neural Video Compression

Zhaoyang Jia[1*] Bin Li[2] Jiahao Li[2] Wenxuan Xie[2] Linfeng Qi[1*] Houqiang Li[1] Yan Lu[2]
[1] University of Science and Technology of China [2] Microsoft Research Asia
{jzy_ustc, qlf324}@mail.ustc.edu.cn, lihq@ustc.edu.cn
{libin, li.jiahao, wenxie, yanlu}@microsoft.com

Abstract

We introduce a practical real-time neural video codec (NVC) designed to deliver high compression ratio, low latency and broad versatility. In practice, the coding speed of NVCs depends on 1) computational costs, and 2) non-computational operational costs, such as memory I/O and the number of function calls. While most efficient NVCs prioritize reducing computational cost, we identify operational cost as the primary bottleneck to achieving higher coding speed. Leveraging this insight, we introduce a set of efficiency-driven design improvements focused on minimizing operational costs. Specifically, we employ implicit temporal modeling to eliminate complex explicit motion modules, and use single low-resolution latent representations rather than progressive downsampling. These innovations significantly accelerate NVC without sacrificing compression quality. Additionally, we implement model integerization for consistent cross-device coding and a module-bank-based rate control scheme to improve practical adaptability. Experiments show our proposed DCVC-RT achieves an impressive average encoding/decoding speed at 125.2/112.8 fps (frames per second) for 1080p video, while saving an average of 21% in bitrate compared to H.266/VTM. The code is available at `https://github.com/microsoft/DCVC`.

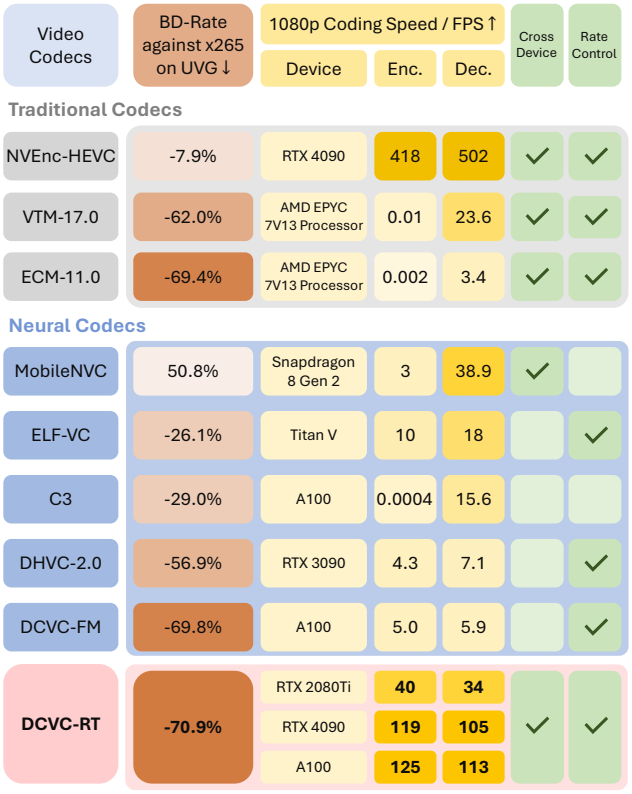


Figure 1. Towards practical real-time neural video codecs (NVCs). Recent advanced NVCs have demonstrated either excellent rate-distortion performance, or improved versatility like integrated cross-device coding consistency or rate-control capabilities. In this paper, we further address the core obstacles of achieving real-time coding to close the last mile toward a practical NVC solution. Our DCVC-RT not only achieves state-of-the-art compression ratio but is also deployable on consumer devices for real-time video coding.

1. Introduction

Neural video codecs (NVCs) have exhibited significant potential in reducing redundancy within video data to achieve higher compression ratios. Since the early work DVC [30], substantial advances [6, 13, 15–17, 19, 25, 27, 31, 32, 39, 42, 45, 51] have been made in enhancing the rate-distortion performance of NVCs. Recent NVCs have surpassed traditional codecs like H.265/HM [2], H.266/VTM [3], and ECM [1]. In this context, compression ratio is no longer the primary bottleneck for NVCs. Instead, the key challenge now lies in how to make NVCs more practical and deployable for real-world applications, to effectively utilize the advantages of such compression ratios.

In response, recent efforts have concentrated on enhanc-

*This work was done when Zhaoyang Jia and Linfeng Qi were full-time interns at Microsoft Research Asia.

†This paper is the outcome of an open-source project started from Dec. 2023.

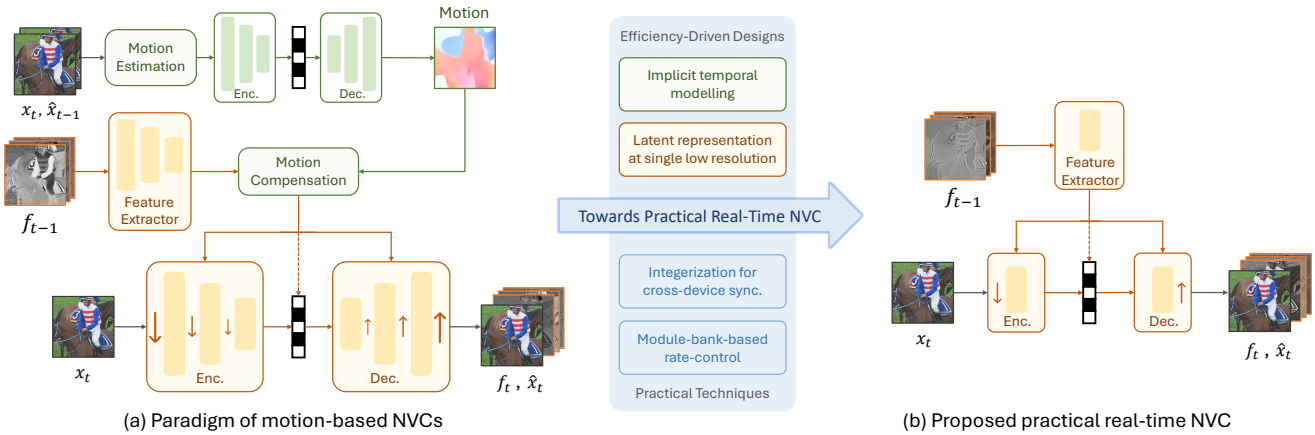


Figure 2. Paradigm shift. To enhance efficiency, we eliminate explicit motion-related modules and adopt implicit temporal modeling. We also propose learning latent representations at a single low resolution, replacing the traditional progressive downsampling approach. Additionally, DCVC-RT supports integerization for cross-device consistency and incorporates a module-bank-based rate-control mechanism.

ing the functionality and versatility of NVCs. Tian et al. [44] introduced auxiliary calibration bitstream transmission to improve cross-device coding accuracy, while MobileCodec [21] and MobileNVC [46] employ deterministic integer calculations to ensure consistent output across different devices. For rate-control functionality, methods like ELF-VC [40], DCVC-FM [25], and DHVC [32, 33] offer controllable rate adjustment within a single model. Zhang et al. [54] developed a rate allocation network for precise bitrate control. These innovations have significantly improved the practicality of NVCs, bringing them closer to real-world deployment.

Despite these advancements, a critical challenge persists for practical NVCs: how to effectively accelerate NVCs for real-time coding? Actually, current NVCs struggle to balance coding speed with rate-distortion performance, leading to a suboptimal **rate-distortion-complexity** trade-off. For instance, while MobileNVC [46] achieves real-time decoding on consumer hardware, its compression ratio is even lower than x264 [4]. C3 [19] provides efficient decoding but relies on time-consuming optimization-based encoding. DHVC-2.0 [32] requires pipelining across multiple GPUs (e.g., 4) to achieve real-time decoding, but its efficiency drops significantly in typical single-GPU environments. Although these methods have significantly accelerated NVCs, real-time coding of 1080p video with high compression ratios on consumer devices remains elusive.

In this paper, we aim to address the core obstacles of achieving real-time coding to close the last mile toward a practical NVC solution. To accelerate NVCs, our first step is to rethink the complexity problem. While most existing research focuses on reducing **computational complexity**, typically measured by the number of multiply-accumulate operations (MACs) during model inference, this alone does not determine the actual coding speed. In practice, many other operations like communication between hardware components, also significantly impact performance. For instance, auto-regressive entropy models [19, 22, 36] require frequent function calls, which incur significant time overhead despite low overall computational cost. Additionally, memory I/O costs of tensors increase with larger tensor sizes, even at the same computational load. We define these factors as the **operational complexity**. Surprisingly, our findings show that high operational overhead, rather than computational cost, is the primary bottleneck in accelerating NVCs.

Based on this insight, we propose a new perspective to accelerate NVCs by reducing operational complexity. In this process, we preserve model capacity by prioritizing more computational capability on the most critical modules while eliminating less essential ones. Firstly, we remove complex motion estimation and compensation process, significantly cutting down the number of components to directly lower the operation frequency. The saved computational capacity is reallocated to frame coding modules to achieve more effective implicit temporal modelling. Additionally, we propose learning latent representations at a single low resolution, i.e., 1/8 of the original image size. Compared to commonly used progressive downsampling method, this approach greatly reduces latent-wise memory I/O overhead while facilitating more effective latent transformations, leading to improved rate-distortion-complexity performance.

With the aforementioned real-time innovations, we further implement model integerization to ensure cross-device consistency and introduce a module-bank-based rate-control technique. Together, these advancements culminate in a practical real-time NVC, DCVC-RT. As shown in Fig. 1, it enables 1080p coding on consumer GPUs like the NVIDIA RTX 2080Ti with an average speed of 40 fps for encoding and 34 fps for decoding. On an NVIDIA A100 GPU, it

reaches an impressive 125 fps for encoding and 113 fps for decoding. Compared to VTM/H.266, our model provides a 21.0% bitrate reduction when using the challenging single intra-frame setting. Additionally, it matches the compression ratio of the advanced DCVC-FM [25] while delivering over 18 times faster coding speed. To the best of our knowledge, DCVC-RT is the first practical NVC to achieve real-time coding with a high compression ratio on consumer hardware.

We summarize the contributions of this paper as follows:

- We investigate the complexity challenges in NVCs and identify operational complexity, rather than computational complexity, as the primary bottleneck.
- Based on this insight, we propose several efficiency-driven designs to reduce operational complexity and enable real-time NVCs. We further enhance the functionality to introduce a practical real-time NVC, DCVC-RT.
- To the best of our knowledge, DCVC-RT is the first real-time NVC to achieve high rate-distortion performance, enabling 1080p real-time coding on consumer hardware with a 21% bitrate reduction compared to VTM/H.266.

2. Related Works

Since the introduction of DVC [30], most research on neural video codecs (NVCs) has focused on enhancing rate-distortion performance. By advancing temporal modeling capabilities [6, 7, 26, 37, 38, 41, 42], improving latent distribution estimation [15, 23, 24, 27], and refining coding paradigms [10, 20, 22, 28, 33, 34], the compression efficiency of NVCs has seen substantial improvement. Recent state-of-the-art NVCs [25, 39] now outperform top traditional codecs like ECM [1]. For NVCs, the primary challenge has shifted from rate-distortion optimization to enhancing functionality and adaptability for real-world deployment.

Real-Time Coding. While real-time coding has been explored in neural image codec [18, 29, 36, 47, 52, 53], it remains relatively underexplored in neural video codecs. Some efforts [40, 44] aim to reduce computational complexity for faster coding but still fall short of achieving real-time 1080p performance. INR-based methods [13, 16, 19] focuses on efficient decoding but requires a time-consuming optimization process for encoding. DHVC-2.0 [32] uses multi-GPU pipelines to achieve real-time throughput for decoding, but falls short in meeting real-time latency requirements and on more common single-GPU devices. MobileNVC [46] achieves real-time decoding throughput but only matches the compression ratio of x264 [4]. In this paper, we present a novel perspective to reduce operational cost rather than computational cost in NVC. Based on it, we introduce several key efficiency-driven techniques to simultaneously achieve 1080p real-time latency with a compression ratio comparable to ECM.

Practical Functionality. For video codecs, maintaining calculation consistency across different devices is a crucial functionality. Typically, this inconsistency arises from non-deterministic floating-point calculations. Ballé et al. [8] and He et al. [14] introduced model integration into neural image codecs to enforce deterministic integer calculations. Similarly, MobileCodec [21] and MobileNVC [46] implement integration in their NVCs to ensure cross-device consistency. Recently, Tian et al. [44] proposed eliminating inconsistency by introducing auxiliary calibration bitstreams. Another important aspect of video codecs is their rate-control capability, particularly in scenarios such as streaming or real-time communication. Zhang et al. [54] developed a rate allocation network to precisely manage bitrate. Other approaches [25, 33, 40] enable continuous, controllable bitrates within a single model, adjusting the model to the target bitrate by manipulating the quantization parameters (qp). However, these methods fail to achieve both cross-device consistency and rate-control capability simultaneously. In contrast, we introduce a practical NVC that support both capabilities, along with real-time coding and a high compression ratio.

3. Rethink the Complexity Problem in NVCs

The primary challenge in the practical application of existing NVCs is their low coding speed. While recent efforts have aimed at reducing computational costs [19, 33, 44], achieving real-time acceleration remains elusive. To address this, we conducted experiments to rethink the complexity problem in NVC acceleration.

In CNNs, **computational complexity** P_{comp} are typically dominated by matrix multiplications, often mitigated by reducing the channel C, as it scales computational complexity by $O(C^2)$. However, our findings reveal that reducing C does not result in the expected quadratic speedup. As illustrated in Fig. 3 (a), speed improves in a more linear fashion as C decreases. This suggests that factors other than computational complexity are limiting the coding speed.

In practice, numerous factors influence coding speed. We identify two key factors: 1) **latent representation size** P_{size}, which primarily influences memory I/O costs of latent tensors; and 2) **number of modules** P_{num}, which affects the total operation counts and the overhead of function calls. These factors primarily influence additional operations beyond hardware computations, which we term **operational complexity**, distinct from computational complexity. We conducted experiments that independently controlled P_{comp}, P_{size} and P_{num} to observe their impact on inference speed. The independent control of each factor is achieved by balancing the number of modules N, channel C and latent resolution $H \times W$. For example, halving C while doubling $H \times W$ maintains a constant P_{size} but reduces P_{comp} due to its quadratic relationship with C.

The results in Fig. 3 (b) reveal several key insights. 1) Operational complexity, rather than computational complexity, is the main speed bottleneck. Reducing computational costs

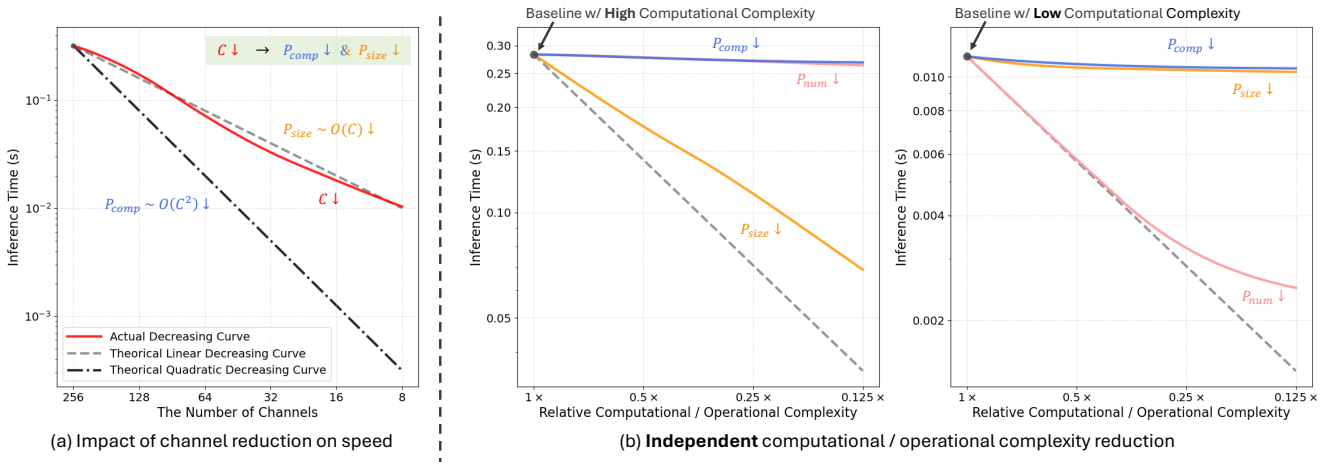


Figure 3. Analysis on computational complexity P_{comp} and operational complexity, including latent representation size P_{size} and number of modules P_{num}. (a) Reducing channels results in a quadratic decrease in P_{comp}, yet inference time decreases almost linearly, indicating that computational cost is not the primary speed bottleneck. (b) We independently reduce one of P_{comp}, P_{size} and P_{num} to identify the main factors affecting time cost. Results show that P_{size} is most critical at high computational complexity, while the P_{num} is more significant at low computational complexity.

without addressing operational factors leads to only marginal improvements in inference time. This also explains why reducing channels results in a linear, rather than quadratic, decrease in time—since the latent size decreases linearly with the number of channels. 2) When computational complexity is high, latent representation size becomes the dominant limiting factor. When computational complexity is low, the number of modules becomes the key bottleneck. This suggests that different parts of the model require different optimization strategies.

These insights offer a new perspective on accelerating NVCs by reducing operational complexity. Typically, lowering computational complexity leads to diminished compression performance. However, since computational complexity is no longer the primary speed bottleneck, we can focus on lowering operational complexity while preserving computational capacity. In our design, we prioritize computational capability to the most critical modules while eliminating less essential ones, which ensures sufficient model capacity and achieves a better rate-distortion-complexity trade-off.

4. Towards Practical Real-Time NVC

4.1. Overview

The framework of the proposed DCVC-RT is illustrated in Fig. 4. To compress current frame x_t, we first patchfy it into $\frac{1}{8}$-scale latents using patch embedding [11]. Then we perform conditional coding [15, 22, 25] in this single low resolution (Section 4.2) to achieve efficient coding. During extracting the temporal context information, DCVC-RT incorporates implicit temporal modeling (Section 4.3) to avoid complex motion-estimation-motion-compensation process. To improve versatility, we introduce a module-bank-based rate-control method (Section 4.4) and enable model integerization (Section 4.5) for cross-device consistency.

4.2. Latents at Single Low Resolution

Originating from the concept of compressive auto-encoders [43], most NVCs progressively downsample latents to reduce the dimension. At each layer, they downsample the latent by half while doubling the number of channels. It enables a comparable computational capacity P_{comp} across layers,

$$P_{comp} \sim O((2C)^2 \cdot H/2 \cdot W/2) = O(C^2 \cdot H \cdot W) \quad (1)$$

while the latent size P_{size} is gradually reduced

$$P_{size} = 2C \cdot H/2 \cdot W/2 = 1/2 \cdot C \cdot H \cdot W \quad (2)$$

In Section. 3, we learn that the latent size can be the main bottleneck for coding speed. From this operational complexity perspective, a question arises: can we learn latents at a single low resolution to eliminate the high operational costs associated with a large P_{size}? To explore this, we directly downsample frames to a single scale using patch embedding and apply conditional coding to compress them at the same scale. Results in Fig. 5 (a) prove the feasibility of this method, where learning single low scale latents notably boosts encoding speed. For example, learning latents at 1/8 scale is about 3.6× faster than progressive downsampling.

While reducing latent scales accelerate model inference, it also impacts rate-distortion performance. Although computational capacity is maintained across scales, the varied receptive field may influence the performance. At a single high 1/2 resolution, the restricted receptive field results in notable performance degradation. However, as scales decrease,

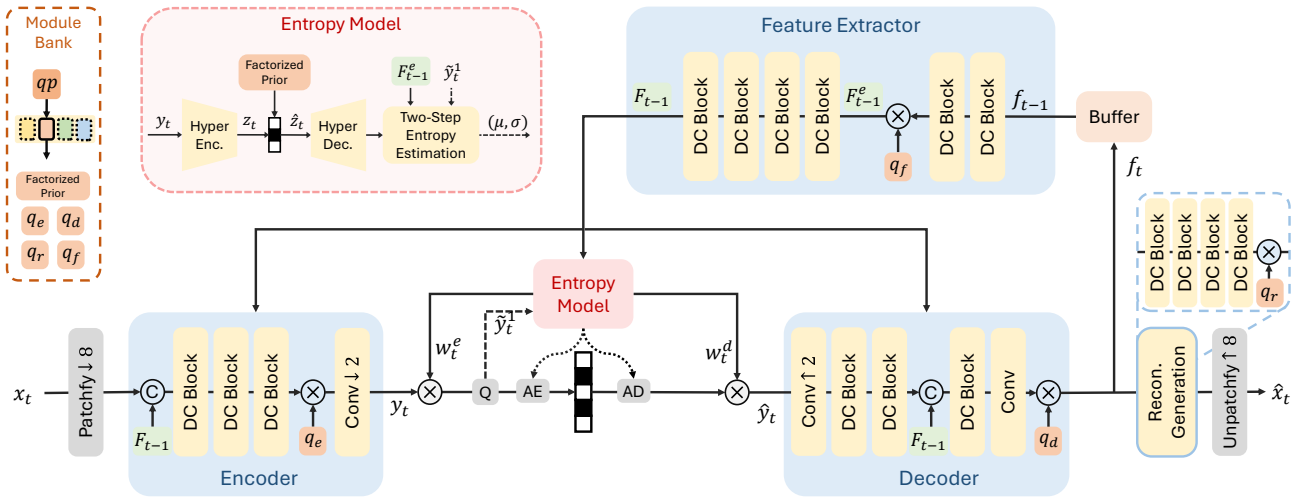


Figure 4. Framework overview. DC Block, Q, AE and AD represent depth-wise convolution block, quantization, arithmetic encoder and decoder, respectively. F_{t-1} and F^e_{t-1} are temporal contexts extracted from previously decoded latent f_{t-1}. Frames are transformed into latents at 1/8 resolution using patch embedding [11], and key modules such as the encoder, decoder, frame extractor, and reconstruction generation operate at this single scale for efficient feature learning. DCVC-RT eliminates explicit motion modeling, resulting in a streamlined design with drastically reduced operational complexity and real-time performance.

the receptive field expands significantly, and at 1/8 scale, it even surpasses the receptive field of progressive downsampling. This extended receptive field is essential for enhancing temporal modeling and reducing temporal redundancy, leading to a comparable BD-Rate of 0.3% under the same model capacity. However, a performance drop is observed at $1/16$ scale. In our model, $1/16$ scale latents with $C = 512$ yields a latent size of $512 \cdot H/16 \cdot W/16 = 2 \cdot H \cdot W$, which is even smaller than the original frame $3 \cdot H \cdot W$. It significantly limits representative capacity and degrades the compression ratio. In contrast, at the $1/8$ scale, using $C = 256$ yields a sufficient latent size of $4 \cdot H \cdot W$. Considering these factors, we adopt 1/8 single-scale latent learning.

4.3. Implicit Temporal Modelling

In video coding, temporal correlation modeling is crucial for effective redundancy reduction. Most existing NVCs achieve this by an explicit motion estimation and motion compensation process. Typically, motion coding needs low computational complexity since motions are simpler and easier for compression. However, we observe that existing motion modules usually use a high number of module layers. For example, we observe that the motion coding branch in [25] exhibits $13\times$ lower computational complexity than the conditional coding branch (74 kMACs per pixel versus 932 kMACs per pixel), despite having up to half as many convolutional layers (123 layers versus 225 layers). As discussed in Section 3, this high number of layers increases operational complexity, becoming the primary speed bottleneck for low-computational-complexity motion modules.

To address this, DCVC-RT adopts implicit temporal modeling, extracting temporal context using a single and simple feature extractor instead of complex motion-based temporal context extraction. Technically, this temporal context is concatenated with the current latent along the channel dimension, allowing the encoder-decoder to process them jointly for redundancy reduction. By eliminating the need for motion estimation and compensation, the number of modules is directly reduced to lower the operation frequency, significantly enhancing the coding speed.

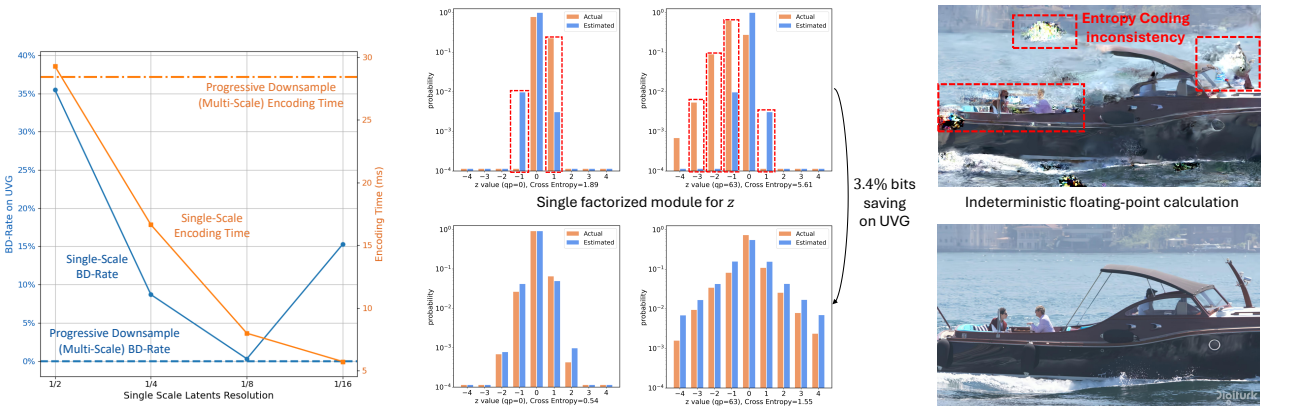
Analysis on Different Motion Contents. We compare implicit and explicit modeling across different motion types for a more comprehensive evaluation.As shown in Tab. 1, implicit modeling slightly improves BD-Rate by 0.4% on small motions while showing a modest 3.2% reduction on large motions. Nonetheless, with a $3.4\times$ faster encoding time, implicit modeling is a more practical solution for real-time applications. Additionally, it surpasses explicit motions in scene changes scenarios, since scene change cannot be effectively modeled by motions. These results highlight its advantages in the rate-distortion-complexity trade-off.

4.4. Module-Bank-Based Rate Control

In DCVC-RT, rate control is achieved through variable-rate coding with dynamic rate adjustment. While existing variable-rate codecs [25, 40] primarily focus on adjusting the distribution of latent y, they typically compress hyper information z using a single factorized prior module. Since z generally accounts for less than 1% of the total bits, it has minimal impact on their performance. However, in DCVC-

Table 1. Ablation study on implicit temporal modelling. We compare it with using explicit motions under different motion contents. The motion range is measured using a pretrained SEA-RAFT [49]. Further details are provided in the supplementary material.

Temporal Modelling	BD-Rate				Encoding Time
	MCL-JCV Average	Large Motion	Small Motion	Scene Change	
Explicit Motions	**0.0%**	**0.0%**	0.0%	0.0%	27.2 ms (3.4×)
Implicit Temporal Modelling	2.1%	3.2%	**–0.4%**	**–4.7%**	8.0 ms (1×)



(a) Ablation study on latent resolutions (b) Effects of rate-adjustment module bank (c) Effects of model intergerization

Figure 5. Analysis of different components. (a) Ablation study on learning latent representations at a single resolution. All models maintain equal computational complexity (MACs) for fairness. (b) Example of probability estimation of z. Using a module bank instead of a single factorized module achieves an average bit savings of 3.4%. (c) Cross platform coding test. We perform encoding on an NVIDIA A100 GPU, while decoding uses an RTX 2080Ti. Model integerization effectively eliminates coding inconsistencies across platforms.

RT, we find that z contributes over 10% bits of y on average, since the absence of motion bits makes z critical in spatial-temporal modeling. In this case, inaccurate distribution estimation for z severely affects overall performance.

To address this, we introduce a rate-adjustment module bank (shown in the top left of Fig. 4). It learns a range of hyperprior modules model varied distributions across different quantization parameter (qp). As shown in Fig. 5 (b), this module bank closely aligns estimated distributions with actual distributions, achieving about 3% bit savings. Extending this approach, we further introduce separate vector banks for different modules (e.g., q_e, q_d, q_f and q_r for encoder, decoder, feature extractor, and reconstruction network, respectively). Each vector bank is designed to learn a set of vectors that adaptively scale the latent representations based on their characteristics, enabling flexible and fine-grained amplitude adjustments. DCVC-RT achieves efficient rate control using this module bank, with the rate-control results provided in the supplementary material.

Moreover, DCVC-RT supports hierarchical quality control by adjusting qp offsets for different frames. Compared to [25] that employs separate feature adaptors for this purpose, our method achieves improved consistency with rate adjustment and enhanced flexibility in practical applications.

4.5. Model integerization

For NVCs, the indeterminism of floating-point calculations can cause inconsistencies when distributing video content. To address this, we implement 16-bit model integerization. This approach enables deterministic integer calculations and ensures consistent output across different devices. More concretely, the equation between a floating-point feature value v_f and an int16 value v_i is as follows

$$v_i = \text{round}(K_1 \cdot v_f) \tag{3}$$

We set $K_1 = 512$, such that the floating-point value 1.0 is mapped to 512 in int16, and given that the valid int16 range is $[-32768, 32767]$, the corresponding range of floating-point values is $[-64.0, 63.998]$. We observe this is sufficiently large to represent the values during model inference. We set the accumulator data type to int32 in convolutional kernels, and no overflow issues have been observed. Besides convolutions and basic arithmetic operations, we adopt a precomputed lookup table to handle the nonlinear Sigmoid function, that maps an arbitrary int16 value to its corresponding output. Through this training-free model integerization, DCVC-RT can perform deterministic integer calculations, ensuring cross-device consistency. An example is shown in Fig. 5 (c), with further results in the supplementary material.

Table 2. BD-Rate (%) comparison in YUV420 colorspace. All frames with intra-period=–1.

	UVG	MCL-JCV	HEVC B	HEVC C	HEVC D	HEVC E	Average	Coding Speed	
								Enc.	Dec.
VTM-17.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.01 fps	23.6 fps
HM-16.25	40.1	48.6	47.6	41.0	34.5	42.8	42.4	0.05 fps	39.6 fps
ECM-11.0	−20.0	−22.1	−22.2	−21.2	−20.4	−17.2	−20.5	0.002 fps	3.4 fps
DCVC-DC	6.5	−4.4	13.1	−3.4	−14.8	90.2	14.5	3.3 fps	4.3 fps
DCVC-FM	−17.6	−8.4	−15.7	−30.2	−37.6	−23.0	−22.1	3.4 fps	4.2 fps
DCVC-FM (fp16)	−16.8	−8.0	−15.4	−30.2	−37.5	−20.2	−21.3	5.0 fps	5.9 fps
DCVC-RT (fp16)	−24.0	−14.8	−16.6	−21.0	−27.3	−22.4	−21.0	125.2 fps	112.8 fps

Note: Some values differ slightly from those in [25] since we use actual BPP instead of estimated BPP.

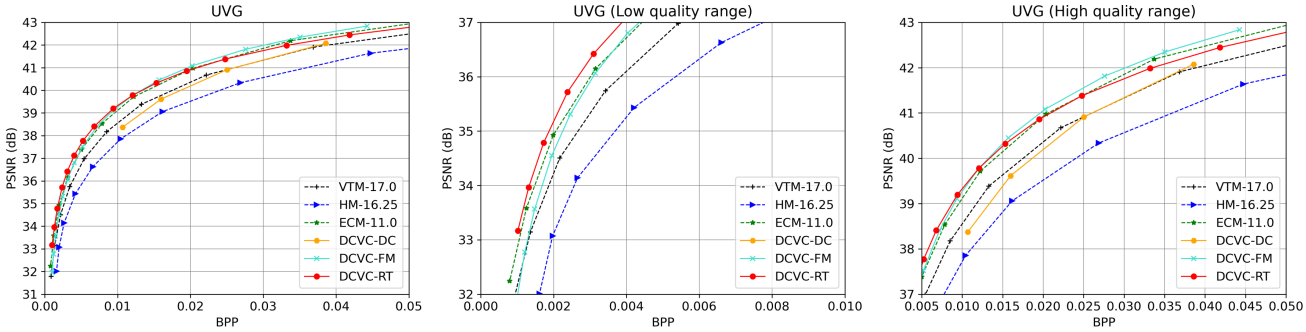


Figure 6. Rate-distortion curves for UVG. All frames are tested in YUV420 colorspace with intra-period=–1. Results on more datasets are in the supplementary materials.

5. Experiments

5.1. Settings

Datasets. We use Vimeo-90k [50] to train DCVC-RT with 7-frame sequences, and process the original Vimeo videos [5] to create longer sequences for fine-tuning by following [25]. We evaluate DCVC-RT on HEVC Class B∼E [12], UVG [35], and MCL-JCV [48].

Test Details. For traditional codecs, we compare with HM [2], VTM [3] and ECM[1], which represent the best H.265, H.266 encoder and the prototype of next generation traditional codec, respectively. Detailed configurations are provided in the supplementary material. For neural codecs, we compare with advanced NVCs including DCVC-DC [24] and DCVC-FM [25]. Following [25], we test all frames with an intra-period of –1 on YUV420 and RGB colorspace. We conduct all tests under low delay conditions. Rate-distortion performance is assessed by the BD-Rate [9]. Additionally, we note that many existing NVCs compare with traditional codecs using the estimated entropy, which is unfair as they overlook header information. In this paper, we ensure a fair comparison by retesting them with actual binary bit-streams that include necessary header information. By default, coding speed is tested on a single NVIDIA A100 GPU with an AMD EPYC 7V13 processor. We measure the average latency across different quantization parameters (qp) on a resolution of 1920×1080.

Training Details. To accommodate variable rates within a single model, we randomly assign different qp between $[0, 63]$ in each training iteration. In a group of 8 pictures, the qp offset is set to $[0, 8, 0, 4, 0, 4, 0, 4]$ for hierarchical quality. We follow [24] to adopt a hierarchical weight setting for the distortion term to support a hierarchical quality structure. The corresponding λ values are interpolated between 1 and 768, following the same method as in [25]. We use the combined distortion loss in both YUV and RGB colorspace [25] to support both colorspace in a single model.

5.2. Comparison Results

In Tab. 2, we present the BD-rate comparison for the YUV420 format under all frame intra period –1 settings. As depicted in the table, DCVC-RT achieves an average 21.0% bits saving compared to VTM, which is slightly better than 20.5% of ECM. It showcases comparable compression ratio to the advanced NVC DCVC-FM with an impressive 25 times faster encoding speed, reaching 125.2 fps for encoding and 112.8 fps for decoding. This demonstrate the superior performance of DCVC-RT in term of rate-distortion-

Table 3. Complexity analysis. The encoding / decoding speed (measured in frames per second, fps) are evaluated across various resolutions and devices, including the NVIDIA A100, NVIDIA A6000, RTX 4090, and RTX 2080 Ti. Average BD-Rate results are presented using VTM as the anchor. MACs are tested on 1080p videos. OOM indicates out-of-memory conditions.

Model	Average BD-Rate	MACs	Params
DCVC-DC	14.5%	2642G	19.8M
DCVC-FM (fp16)	–21.3%	2642G	18.3M
DCVC-RT (fp16)	–21.0%	385G	20.7M
DCVC-RT (int16)	–18.3%	385G	20.7M

(a) Computational complexity and BD-Rate.

Model	A100	A6000	4090	2080Ti
DCVC-DC	3.3 / 4.3	1.7 / 2.2	2.3 / 2.9	0.8 / 1.4
DCVC-FM (fp16)	5.0 / 5.9	3.1 / 3.8	3.7 / 4.4	1.9 / 2.3
DCVC-RT (fp16)	125.2 / 112.8	70.4 / 63.8	118.8 / 105.3	39.5 / 34.1
DCVC-RT (int16)	28.3 / 20.9	23.4 / 17.5	52.3 / 38.8	18.4 / 13.4

(b) Coding speed on 1920×1080 videos.

Model	A100	A6000	4090	2080Ti
DCVC-DC	0.8 / 1.0	0.4 / 0.5	OOM	OOM
DCVC-FM (fp16)	1.0 / 1.2	0.6 / 0.7	OOM	OOM
DCVC-RT (fp16)	35.5 / 29.5	18.5 / 16.2	29.9 / 26.5	11.6 / 9.9
DCVC-RT (int16)	7.3 / 5.2	6.1 / 4.4	12.5 / 9.5	4.4 / 3.2

(c) Coding speed on 3840×2160 videos.

Model	A100	A6000	4090	2080Ti
DCVC-DC	6.5 / 7.9	3.5 / 4.3	5.5 / 6.7	2.1 / 2.9
DCVC-FM (fp16)	8.5 / 9.4	5.9 / 6.6	9.3 / 10.4	4.0 / 4.7
DCVC-RT (fp16)	173.9 / 149.2	147.3 / 132.5	225.1 / 185.2	73.3 / 67.0
DCVC-RT (int16)	51.7 / 39.2	49.5 / 38.1	105.2 / 81.1	37.0 / 25.8

(d) Coding speed on 1280×720 videos.

complexity trade-off. In the RGB colorspace, DCVC-RT achieves a 14.0% bits saving compared to VTM, closely matching the 15.8% savings of DCVC-FM. Detailed results are provided in the supplementary material.

Fig. 6 presents the rate-distortion curve on UVG. DCVC-RT showcases better performance with VTM across the entire quality range. Particularly in the low-quality range (< 0.02 bpp), DCVC-RT exhibits the best performance. However, there is a performance drop in the high-quality range. This drop can be attributed to the lightweight model design adopted in DCVC-RT, resulting in reduced model capability compared to larger models. Notably, this drop mainly occurs above 40 dB, where human vision struggles to distinguish between different qualities. In the supplementary material, we further examine the compression performance of DCVC-RT as model capacity increases. Our large model achieves the highest compression ratio across all bitrate ranges while maintaining real-time performance.

5.3. Complexity Analysis

Tab. 3 presents the complexity analysis. Compared to DCVC-DC and DCVC-FM, DCVC-RT achieves significantly lower computational complexity while maintaining a comparable compression ratio. Coding speed is evaluated across multiple input resolutions and GPU devices, consistently demonstrating at least a $20\times$ speed improvement. On the A100 GPU, DCVC-RT (fp16) reaches real-time 4K 30fps coding, while on consumer-grade devices like the RTX 2080 Ti, it achieves 1080p 30fps coding. These results highlight the efficiency of DCVC-RT across diverse conditions.

5.4. Integerization Results

Our model supports 16-bit integer calculations. As shown in Tab. 3, our integerization strategy introduces minimal impact on compression performance, with DCVC-RT (int16) still outperforming VTM by 18.3%. Dataset-specific BD-Rate results are in the supplementary material. In terms of coding speed, DCVC-RT (int16) achieves 1080p 30 fps coding on an RTX 4090 and 720p 24 fps coding on an RTX 2080Ti.

However, we observe a significant slowdown in coding speed when using int16 mode compared to fp16. This is mainly because most modern GPUs lack dedicated optimization for int16 operations. This difference is particularly pronounced on the A100 GPU, where highly optimized Tensor Cores make fp16 processing over four times faster than int16. Although int16 mode theoretically has the potential to enable faster inference than fp16, we anticipate that future hardware developments and engineering will help bridge this performance gap.

6. Conclusion and Limitation

In this paper, we propose a practical, real-time neural video codec (NVC) focused on high compression ratio, low latency, and broad versatility. By analyzing the complexity of NVCs, we identify operational cost, rather than computational cost, as the primary bottleneck to coding speed. Based on this insight, we employ implicit temporal modeling and a single low-resolution latent representation, which significantly accelerates processing without compromising compression quality. Additionally, we introduce model integerization for consistent cross-device coding and a module-bank-based rate control scheme to enhance practical adaptability. As far as we known, DCVC-RT is the first NVC achieving 110 fps coding on 1080p video with a 21% bitrate savings compared to H.266/VTM. DCVC-RT serves as a notable landmark in the journey of NVC evolution.

While DCVC-RT supports int16 mode, its coding speed remains slower than fp16 due to limited hardware optimization for int16 inference. In the future, we hope this can be solved by further hardware optimization and engineering.

References

[1] ECM. https://vcgit.hhi.fraunhofer.de/ecm/ECM.

[2] HM. https://vcgit.hhi.fraunhofer.de/jvet/HM.

[3] VTM. https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM.

[4] FFmpeg. https://www.ffmpeg.org/.

[5] Original Vimeo links. https://github.com/anchen1011/toflow/blob/master/data/original_vimeo_links.txt.

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

[7] David Alexandre, Hsueh-Ming Hang, and Wen-Hsiao Peng. Hierarchical B-frame Video Coding Using Two-Layer CANF without Motion Coding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10249–10258, 2023.

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

[9] Gisle Bjontegaard. Calculation of average PSNR differences between RD-curves. *VCEG-M33*, 2001.

[10] Zhenghao Chen, Lucas Relic, Roberto Azevedo, Yang Zhang, Markus Gross, Dong Xu, Luping Zhou, and Christopher Schroers. Neural Video Compression with Spatio-Temporal Cross-Covariance Transformers. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 8543–8551, 2023.

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*, 2021.

[12] D Flynn, K Sharman, and C Rosewarne. Common Test Conditions and Software Reference Configurations for HEVC Range Extensions, document JCTVC-N1006. *Joint Collaborative Team Video Coding ITU-T SG*, 16.

[13] Ge Gao, Ho Man Kwan, Fan Zhang, and David Bull. Pnvc: Towards practical inr-based video compression. *arXiv preprint arXiv:2409.00953*, 2024.

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

[15] Yung-Han Ho, Chih-Peng Chang, Peng-Yu Chen, Alessandro Gnutti, and Wen-Hsiao Peng. CANF-VC: Conditional augmented normalizing flows for video compression. In *European Conference on Computer Vision*, pages 207–223. Springer, 2022.

[16] Zhihao Hu and Dong Xu. Complexity-guided slimmable decoder for efficient deep video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14358–14367, 2023.

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

[18] Zhaoyang Jia, Jiahao Li, Bin Li, Houqiang Li, and Yan Lu. Generative latent coding for ultra-low bitrate image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26088–26098, 2024.

[19] Hyunjik Kim, Matthias Bauer, Lucas Theis, Jonathan Richard Schwarz, and Emilien Dupont. C3: High-performance and low-complexity neural compression from a single image or video. pages 9347–9358, 2024.

[20] Théo Ladune, Pierrick Philippe, Wassim Hamidouche, Lu Zhang, and Olivier Déforges. Conditional Coding for Flexible Learned Video Compression. In *Neural Compression: From Information Theory to Applications – Workshop @ ICLR 2021*, 2021.

[21] 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.

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

[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.

[24] Jiahao Li, Bin Li, and Yan Lu. Neural video compression with diverse contexts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22616–22626, 2023.

[25] Jiahao Li, Bin Li, and Yan Lu. Neural Video Compression with Feature Modulation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 17-21, 2024*, 2024.

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

[27] Bowen Liu, Yu Chen, Rakesh Chowdary Machineni, Shiyu Liu, and Hun-Seok Kim. 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*, pages 18487–18496, 2023.

[28] Jerry Liu, Shenlong Wang, Wei-Chiu Ma, Meet Shah, Rui Hu, Pranaab Dhawan, and Raquel Urtasun. Conditional entropy coding for efficient video compression. In *European Conference on Computer Vision*, pages 453–468. Springer, 2020.

[29] Jinming Liu, Heming Sun, and Jiro Katto. Learned image compression with mixed transformer-cnn architectures. In

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 14388–14397, 2023.

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

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

[32] Ming Lu, Zhihao Duan, Wuyang Cong, Dandan Ding, Fengqing Zhu, and Zhan Ma. High-Efficiency Neural Video Compression via Hierarchical Predictive Learning. *arXiv preprint arXiv:2410.02598*, 2024.

[33] Ming Lu, Zhihao Duan, Fengqing Zhu, and Zhan Ma. Deep Hierarchical Video Compression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8859–8867, 2024.

[34] Fabian Mentzer, George D Toderici, David Minnen, Sergi Caelles, Sung Jin Hwang, Mario Lucic, and Eirikur Agustsson. VCT: A Video Compression Transformer. In *Advances in Neural Information Processing Systems*, pages 13091–13103, 2022.

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

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

[37] Reza Pourreza, Hoang Le, Amir Said, Guillaume Sautiere, and Auke Wiggers. Boosting neural video codecs by exploiting hierarchical redundancy. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5355–5364, 2023.

[38] Linfeng Qi, Jiahao Li, Bin Li, Houqiang Li, and Yan Lu. Motion Information Propagation for Neural Video Compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6111–6120, 2023.

[39] Linfeng Qi, Zhaoyang Jia, Jiahao Li, Bin Li, Houqiang Li, and Yan Lu. Long-term temporal context gathering for neural video compression. In *European Conference on Computer Vision*, pages 305–322. Springer, 2024.

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

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

[42] Yibo Shi, Yunying Ge, Jing Wang, and Jue Mao. AlphaVC: High-performance and efficient learned video compression. In *European Conference on Computer Vision*, pages 616–631. Springer, 2022.

[43] L. Theis, W. Shi, A. Cunningham, and F. Huszár. Lossy image compression with compressive autoencoders. 2017.

[44] Kuan Tian, Yonghang Guan, Jinxi Xiang, Jun Zhang, Xiao Han, and Wei Yang. Towards real-time neural video codec for cross-platform application using calibration information. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 7961–7970, 2023.

[45] Ties Van Rozendaal, Iris AM Huijben, and Taco S Cohen. Overfitting for fun and profit: Instance-adaptive data compression. *arXiv preprint arXiv:2101.08687*, 2021.

[46] Ties van Rozendaal, Tushar Singhal, Hoang Le, Guillaume Sautiere, Amir Said, Krishna Buska, Anjuman Raha, Dimitris Kalatzis, Hitarth Mehta, Frank Mayer, et al. MobileNVC: Real-time 1080p Neural Video Compression on a Mobile Device. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4323–4333, 2024.

[47] Guo-Hua Wang, Jiahao Li, Bin Li, and Yan Lu. EVC: Towards Real-Time Neural Image Compression with Mask Decay. In *International Conference on Learning Representations*, 2023.

[48] 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.

[49] Yihan Wang, Lahav Lipson, and Jia Deng. SEA-RAFT: Simple, efficient, accurate raft for optical flow. In *European Conference on Computer Vision*, pages 36–54. Springer, 2025.

[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.

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

[52] Yibo Yang and Stephan Mandt. Computationally-efficient neural image compression with shallow decoders. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 530–540, 2023.

[53] Xinjie Zhang, Xingtong Ge, Tongda Xu, Dailan He, Yan Wang, Hongwei Qin, Guo Lu, Jing Geng, and Jun Zhang. Gaussianimage: 1000 fps image representation and compression by 2d gaussian splatting. In *European Conference on Computer Vision*, pages 327–345. Springer, 2024.

[54] Yiwei Zhang, Guo Lu, Yunuo Chen, Shen Wang, Yibo Shi, Jing Wang, and Li Song. Neural Rate Control for Learned Video Compression. In *The Twelfth International Conference on Learning Representations*, 2023.