國立臺灣大學電機資訊學院電信工程學研究所

碩士論文

Graduate Institute of Communication Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master's Thesis

國立臺灣大學碩博士畢業論文模版

National Taiwan University (NTU) Thesis/Dissertation Template in LaTeX

陳祈安

Chi-An Chen

指導教授: 丁建均 博士

Advisor: Jian-Jiun Ding, Ph.D.

中華民國 115 年 1 月

January 2026

國立臺灣大學碩士學位論文(初稿)
口試委員會審定書

MASTER'S THESIS (DRAFT) ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

國立臺灣大學碩博士畢業論文模版

National Taiwan University (NTU)
Thesis/Dissertation Template in LaTeX

本論文係朱雁丞 (R10921008) 在國立臺灣大學電機工程學系完成之碩士學位論文(初稿)，於民國 111 年 8 月 14 日 承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Department of Electrical Engineering on 14/08/2022 have examined a Master's Thesis (Draft) entitled above presented by Yen-Cheng Chu R10921008 candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

(指導教授 Advisor)		

系主任 Director: _____

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Background

Modern digital life and network communication are fundamentally built upon efficient data compression technologies. With the explosive growth of global data volume—particularly the rapid increase in multimedia content—compression techniques have become increasingly critical for maintaining Quality of Service (QoS).

Video content has now become the dominant source of network traffic. According to the Ericsson Mobility Report[1], video traffic is projected to account for approximately 76% of all mobile data traffic by the end of 2025. This trend is driven not only by traditional video streaming platforms such as YouTube, Netflix, and TikTok, but has also been further intensified by the widespread adoption of short-form video sharing on social media, real-time video conferencing, and the growing prevalence of high-quality images and videos generated by artificial intelligence (AI). Consequently, designing high-performance video compression systems that can deliver higher visual quality under limited bandwidth budgets has become an increasingly urgent challenge.

Although conventional video coding standards—such as H.264/AVC[2], H.265/HEVC[3], and Versatile Video Coding (VVC)[4]—have achieved remarkable success and have been continuously optimized over the past decades, Neural Video Coding (NVC) has emerged in recent years as a promising alternative. By leveraging the strong nonlinear modeling capability of neural networks, NVC has demonstrated significant potential to surpass traditional hybrid video coding frameworks in terms of compression efficiency [5].

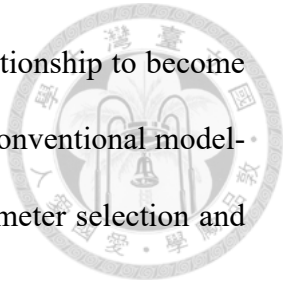
With the rapid advancement of generative models in visual signal representation and processing, their impact has extended beyond academic research to international video coding standardization. Major standardization bodies, including the Joint Video Experts Team (JVET) and ISO/IEC MPEG, have actively explored neural network-based approaches to visual coding. A representative example is JPEG AI[6], a joint standardization effort between ISO/IEC JPEG and ITU-T, which adopts learning-based techniques for image compression while supporting downstream computer vision tasks. These efforts highlight neural video coding as a recognized and promising direction for future visual coding systems.

1.2 Problem Statement

Despite the rapid progress of neural video coding (NVC), rate control (RC) remains a critical component for its practical and commercial deployment. In real-time applications such as video conferencing and live streaming, bitrate is the primary physical resource constraint, and unstable rate control may lead to buffer overflow or underflow, resulting in latency, frame drops, inefficient bandwidth utilization, and degraded Quality of Service (QoS). Therefore, a stable and reliable rate control mechanism is indispensable for real-time NVC systems.

Conventional rate control methods in hybrid video coding frameworks are typically derived from rate–distortion optimization (RDO) and rely on simplified and approximately stationary models that relate bitrate to the quantization parameter[7]. Although effective for traditional codecs, these assumptions no longer hold for modern coding systems with increasingly flexible and adaptive coding structures.

In NVC, end-to-end learned representations introduce highly nonlinear, frame-level,



and content-dependent rate characteristics, causing the Rate–QP relationship to become non-stationary and difficult to model. As a result, directly applying conventional model-based rate control methods often leads to unstable quantization parameter selection and inaccurate bitrate estimation in practical deployment.

In summary, there is currently a lack of real-time rate control solutions that simultaneously achieve high accuracy, low computational complexity, and robust adaptability to modern NVC architectures. Addressing this challenge is essential for enabling the practical deployment of neural video coding systems.

1.3 Motivation

Throughout the development of NVC, encoding and decoding speed has long been one of the primary bottlenecks limiting practical deployment. The emergence of DCVC-RT [8], however, marks an unprecedented milestone. DCVC-RT successfully achieves a favorable balance between computational efficiency and compression performance. In most scenarios, it surpasses conventional codecs in both speed and coding efficiency, and even outperforms experimental software codecs such as ECM[9] that have yet to be officially released. This breakthrough effectively removes the long-standing perception that NVC is impractical due to excessive computational complexity, thereby enabling realistic prospects for commercial deployment.

Under this context, efficient rate control has become the “last mile” toward large-scale deployment of NVC systems. Although DCVC-RT has demonstrated the coding potential of modern NVC architectures, research on rate control specifically tailored to such state-of-the-art (SOTA) NVC models remains limited. Most existing studies [10, 11, 12] focus on earlier NVC architectures and rely on extensions of traditional mathematical

models. Moreover, due to the rapid evolution of NVC architectures in recent years, it has been difficult to establish unified and generalizable experimental frameworks.

Given the superior performance and representative nature of DCVC-RT, investigating its Rate–QP relationship and designing a dedicated, lightweight rate control mechanism offers substantial academic and industrial value. Accordingly, this thesis focuses on DCVC-RT as a practically viable NVC model and aims to address the core rate control challenges that hinder the real-world deployment of neural video coding systems.

1.4 Primary Contributions

This thesis aims to address the aforementioned challenges by proposing an efficient and low-complexity real-time rate control framework for modern NVC architectures. The main contributions are summarized as follows:

1. **Adaptive State-Space Modeling for NVC Rate Control**

    - This work proposes a low-complexity adaptive state-space model specifically designed to describe and predict the nonlinear Rate–QP relationship exhibited by modern NVC architectures such as DCVC-RT.
    - To the best of our knowledge, this study is the first to introduce Recursive Least Squares (RLS)[13] into the rate control modeling of NVC systems, enabling gradient-free and highly accurate real-time bitrate estimation and control.

2. **Improved Performance–Complexity Trade-off**

    - Compared with conventional rate control methods and algorithms designed for earlier NVC architectures, the proposed RLS-based system achieves a more

favorable balance among bitrate accuracy, visual quality stability, and computational efficiency.

- The proposed RLS framework incurs very low computational overhead, with approximately $O(N^2)$ complexity, where N denotes the number of parameters in the state-space model. This fully satisfies the low-latency requirements of real-time video encoding.

3. **Comprehensive Parameter Analysis and Experimental Transparency**

   - This study provides a comprehensive analysis of the core algorithms and key parameters within the proposed rate control framework.
   - Experimental details that are often insufficiently documented in existing NVC literature—such as initialization strategies and experimental configurations—are explicitly discussed. The impact of these parameters is systematically evaluated, providing more reliable baselines and references for future research on NVC rate control.

1.5 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 introduces the fundamentals of video compression, including traditional hybrid coding frameworks and neural video compression (NVC). Chapter 3 reviews related work on state-of-the-art NVC architectures and existing rate control methods, and discusses their limitations. Chapter 4 presents the proposed adaptive state-space rate control framework based on recursive least squares (RLS). Chapter 5 evaluates the proposed method through experimental comparisons and ablation studies. Finally, Chapter 6 concludes the thesis and outlines directions for future work.

Chapter 2

Fundamentals

2.1 Video Compression Basics

2.1.1 Information Theory – Entropy

To understand the theoretical limits of video compression, we introduce the fundamental concepts established by Claude Shannon [14]. The definitions in this section are primarily summarized from [15]. The core idea revolves around quantifying ”information” and establishing a lower bound for the number of bits required to represent data.

Self-Information and Entropy

Shannon defined a quantity called *self-information* based on probability. Let A be an event with a probability $P(A)$. The self-information is defined as $i(A) = -\log_2 P(A)$. This aligns with the intuition that low-probability events (e.g., rare pixel values) contain high information, while high-probability events contain little.

For a source generating a sequence of independent and identically distributed (i.i.d) symbols X, the average self-information is known as the **Entropy** (H):

$$H(S) = -\sum P(X) \log_2 P(X) \tag{2.1}$$

Relevance to Compression

The significance of entropy lies in Shannon’s Source Coding Theorem. It demonstrates that the best a lossless compression scheme can do is to encode the source with an

average number of bits equal to its entropy.

2.1.2 Redundancy

Raw video data require an extremely large amount of bandwidth. Efficient compression exploits inherent redundancies: spatial, temporal, and coding redundancy, along with perceptual redundancy in color space.

Spatial Redundancy

Spatial redundancy arises from the strong statistical correlation between neighboring pixels in natural images. As illustrated in Fig. 2.1, image content can generally be categorized into flat regions, edges, and textures. Flat regions (e.g., the smooth background) exhibit the highest redundancy with minimal pixel variance, whereas edges (e.g., facial boundaries) and textures (e.g., hair strands) contain high-frequency information.

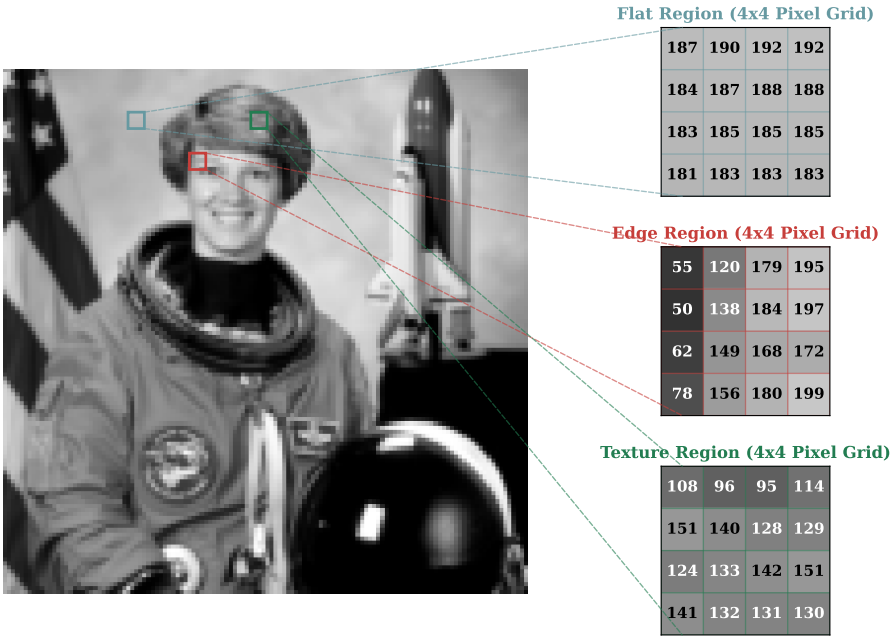


Figure 2.1: Spatial redundancy in the 'Astronaut' image: flat (background), edge (facial features), and texture (hair) regions.

To exploit this redundancy, **Differential Pulse-Code Modulation (DPCM)** is com-

monly employed. DPCM operates by predicting the current pixel x_n based on its reconstructed neighbor x_{n-1} and encoding the difference, known as the residual ($d_n = x_n - x_{n-1}$). As demonstrated in Fig. 2.2, while the original pixel intensities spans the full dynamic range (top), the prediction residuals form a **peaked Laplacian-like probability distribution** centered at zero (bottom). This energy compaction significantly reduces the entropy, thereby facilitating more efficient entropy coding.

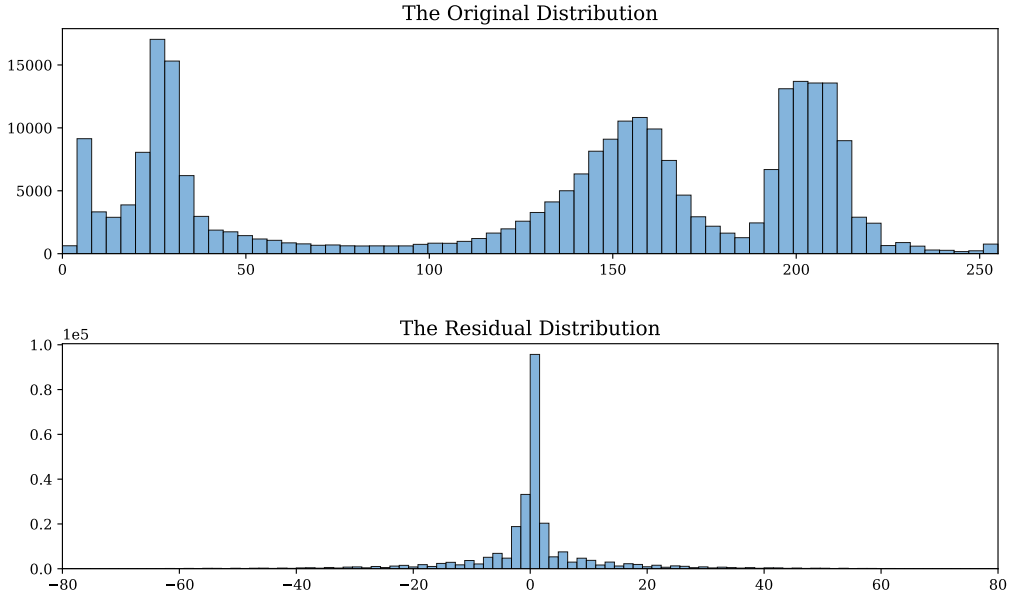


Figure 2.2: Statistical comparison: original pixel intensities distribution (top) vs. prediction residuals distribution (bottom).

Temporal Redundancy

Successive video frames, denoted as I_t and I_{t+1}, typically exhibit strong statistical correlation, a phenomenon known as **temporal redundancy**. To eliminate this redundancy, modern video coding standards employ **inter-frame prediction**, which consists of two core components:

1. **Motion Estimation (ME):** The encoder partitions the current frame into blocks and searches for the optimal matching block in the reference frame. The spatial displacement is encoded as a **Motion Vector (MV)**.

2. **Motion Compensation (MC):** The reference frame is warped using the derived MVs to synthesize a prediction of the current frame.

The coding efficiency is largely determined by the **residual**, defined as the difference between the actual current frame and the prediction. Fig. 2.3 visualizes this process using a sample from the Vimeo-90k dataset [16]. Due to camera motion, the direct difference between frames results in significant high-frequency energy along object boundaries. In contrast, applying motion compensation effectively aligns the temporal features, reducing the residual to a sparse, noise-like signal.

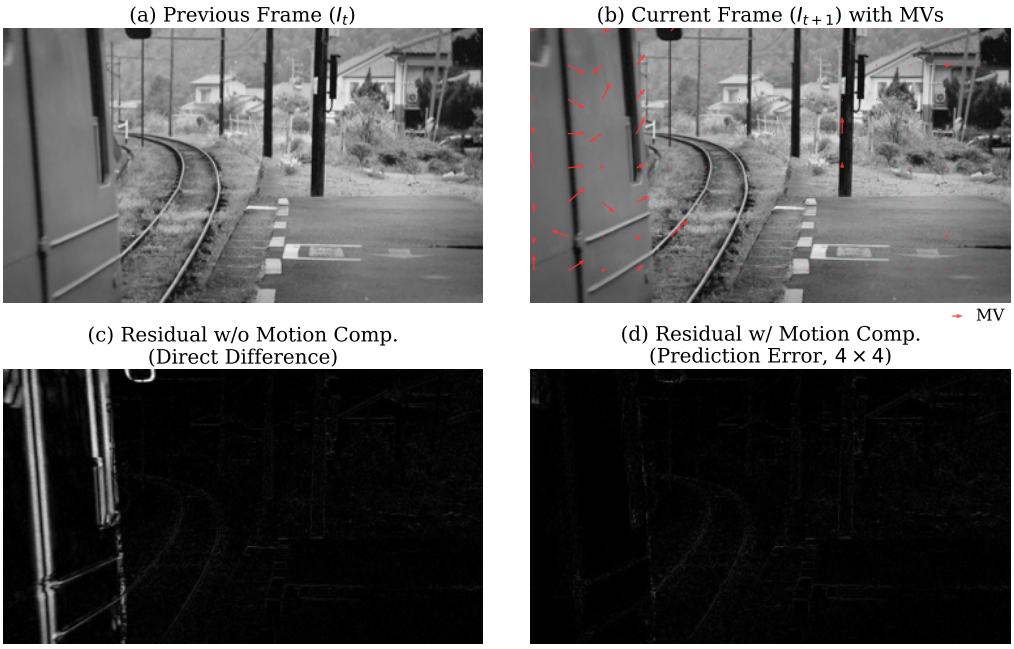


Figure 2.3: Temporal redundancy reduction: motion estimation (top) and residual comparison (bottom).

Coding Redundancy

Coding redundancy occurs when the average codeword length exceeds the theoretical minimum limit determined by the source entropy. While fixed-length coding assigns equal bits to all symbols regardless of their frequency, **Variable-Length Coding (VLC)**, such as **Huffman coding** [17], minimizes redundancy by assigning shorter codewords to high-

probability symbols (e.g., symbol 'F' in Fig.2.4) and longer codewords to rare ones. As illustrated in Fig.2.4, the Huffman algorithm yields an average code length of $\bar{L} \approx 2.24$ bits, which is significantly lower than the fixed-length approach ($L = 3$ bits) and closely approximates the source entropy ($H \approx 2.22$ bits).

Symbol	A	B	C	D	E	F
Frequency	5	9	12	13	16	45
Code	1100	1101	100	101	111	0

$$\bar{L} = \frac{14 \cdot 4 + 41 \cdot 3 + 45 \cdot 1}{100} \approx 2.24$$

Fixed length, 6 symbols $\mathrm{L} = \lceil \log_2(6) \rceil = 3$

$$\textit{Entropy}: \quad H = -\sum_i p_i \log_2(p_i) \approx 2.22$$

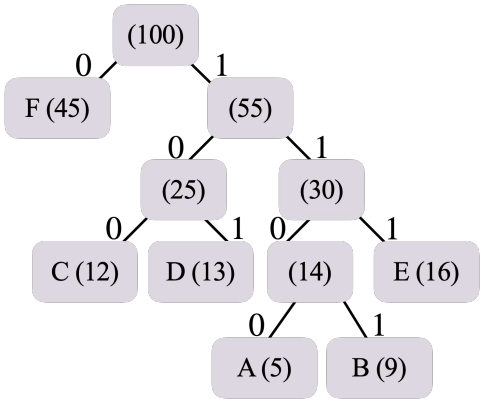


Figure 2.4: Huffman tree for a six-symbol source with average code length $\bar{L} \approx 2.24$.

Color Space Representation

While video signals are typically captured in the RGB domain, the Red, Green, and Blue channels exhibit high statistical correlation. To remove this redundancy, video coding standards transform RGB into the **YUV color space** (specifically YCbCr), which separates the signal into **Luminance (Y)** and **Chrominance (U, V)**. This transformation exploits a fundamental property of the **Human Visual System (HVS)**: the eyes are significantly more sensitive to brightness changes than to color variations. Consequently, the chroma components can be spatially subsampled (e.g., **4:2:2** or **4:2:0**) to reduce the bitrate. As demonstrated in Fig. 2.5, the structural information is concentrated in the Y channel, while the U and V channels appear smoother. The final reconstruction after 4:2:2 subsampling confirms that this reduction in color resolution results in negligible perceptual degradation.

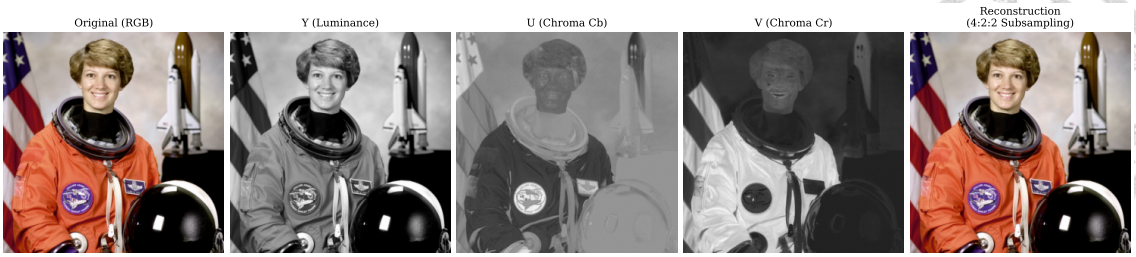


Figure 2.5: Decomposition of the 'Astronaut' image into YUV components and the reconstructed output.

2.1.3 The Rate-Distortion (R-D) Trade-off

While entropy sets the theoretical limit for lossless compression, practical video coding relies on **lossy compression** to satisfy bandwidth constraints. This necessitates a fundamental trade-off: minimizing Bitrate (R) while minimizing Distortion (D).

Quantization and Reconstruction

In both hybrid frameworks and Neural Video Coding (NVC), **Quantization** serves as the decisive mechanism to govern this trade-off. It reduces the precision of transform coefficients (or latent features) to reduce information entropy. As illustrated in Fig.2.6, the process typically involves an element-wise division by a quantization step or matrix (Q), followed by a rounding operation.

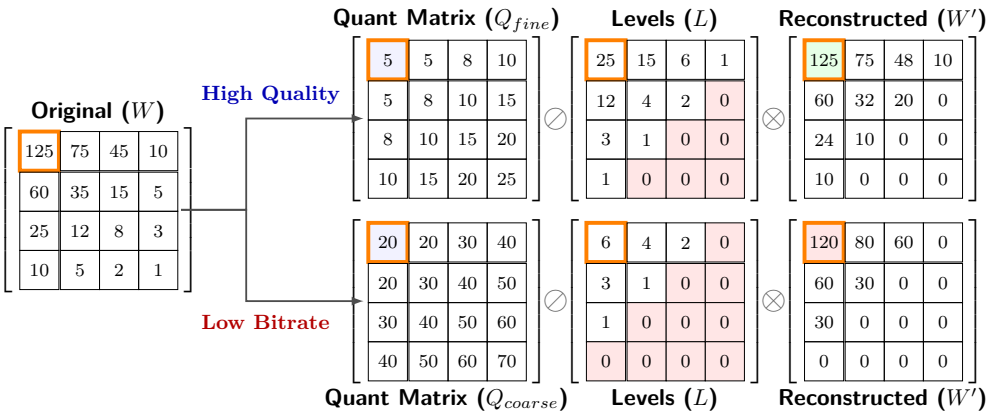
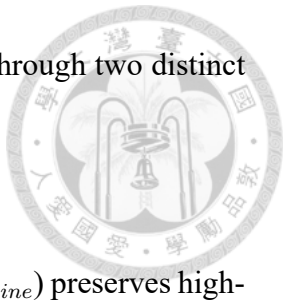


Figure 2.6: The mechanics of quantization. Coarse quantization (bottom path) reduces bitrate at the cost of reconstruction fidelity.



The figure demonstrates the impact of quantization granularity through two distinct paths:

- **High Quality Path (Top):** Using a fine quantization matrix (Q_{fine}) preserves high-frequency details. The reconstruction is nearly lossless (e.g., the DC coefficient is restored as 125), but the resulting symbol matrix retains high entropy, requiring more bits to encode.

- **Low Bitrate Path (Bottom):** Using a coarse quantization matrix (Q_{coarse}) forces many high-frequency coefficients to zero (indicated in red). While this sparsity significantly reduces the bitrate, it introduces irreversible quantization error. As highlighted in the reconstructed matrix (W'), the DC coefficient is distorted ($120 \neq 125$), representing the D term in the R-D trade-off.

Lagrangian Rate-Distortion Optimization

The relationship between rate and distortion is empirically convex, as shown in the R-D curves in Fig.2.7. Initially, a small increase in bitrate yields significant quality gains, but the returns eventually diminish.

To automate the selection of the optimal operating point along these curves, modern encoders employ **Rate-Distortion Optimization (RDO)** using the method of Lagrange multipliers. The system seeks to minimize a joint cost function J:

$$\min J = D + \lambda R \tag{2.2}$$

Here, the Lagrange multiplier λ acts as a trade-off parameter. A larger λ imposes

a heavy penalty on rate, pushing the encoder towards the ”Low Bitrate” path, whereas a smaller λ prioritizes reconstruction fidelity.

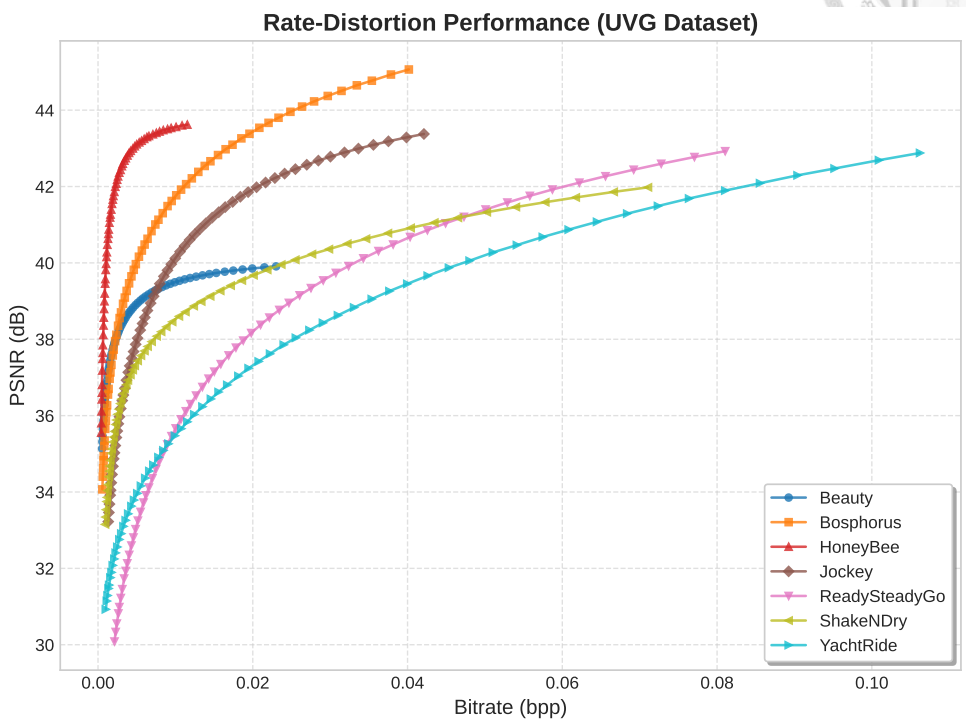


Figure 2.7: Rate-Distortion (R-D) performance curves evaluated on the UVG dataset, illustrating the trade-off between quality (PSNR) and bitrate (bpp).

2.2 From Traditional Standards to Neural Video Coding

2.2.1 The Generalized Hybrid Coding Pipeline

Modern video compression standards, such as H.264/AVC [2] and H.265/HEVC [3], have dominated the industry for decades. These standards rely on the classic *hybrid coding architecture*, which combines predictive coding (to exploit temporal redundancy) and transform coding (to exploit spatial redundancy). Interestingly, the emergence of Neural Video Coding (NVC) does not necessarily discard this proven architecture. Instead, early pioneering works, such as the Deep Video Compression (DVC) framework by Lu et al. [18], propose an end-to-end learning framework that maintains a one-to-one correspondence with traditional standards.

Fig. 2.8 illustrates this architectural evolution. On the left, the traditional framework consists of hand-crafted modules: block-based motion estimation, integer transform derived from discrete cosine transform (DCT), quantization, and entropy coding. Although each module is well-designed, they are optimized separately, which may not lead to the global optimum for the entire system.

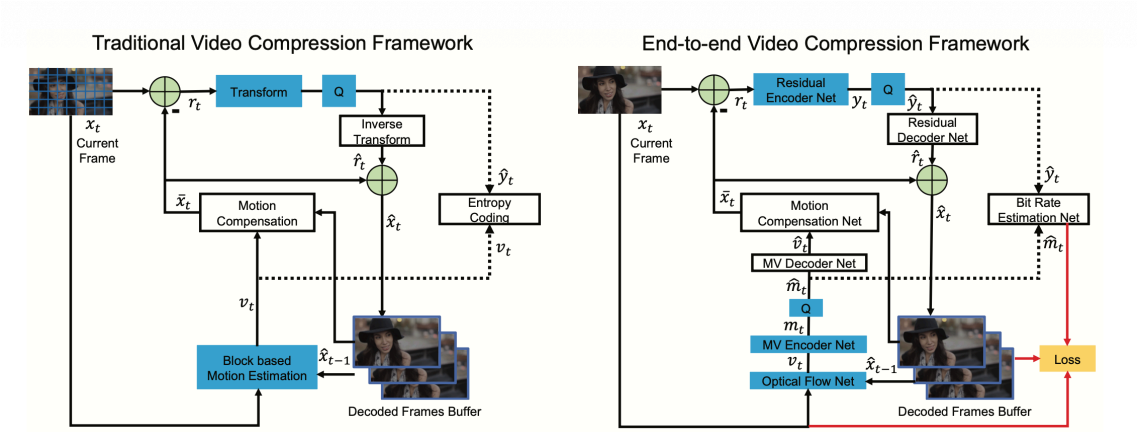


Figure 2.8: Comparison between the traditional predictive coding architecture (left) and the end-to-end Deep Video Compression (DVC) framework (right) [18].

On the right side of Fig. 2.8, the DVC framework replaces these key components with neural networks:

- **Motion Estimation & Compensation:** The traditional block-based search is replaced by an Optical Flow Net and a Motion Compensation Net to perform pixel-wise prediction.
- **Transform & Inverse Transform:** The linear transform is replaced by non-linear Residual Encoder and Decoder networks, which utilize the powerful representation ability of Convolutional Neural Networks (CNNs) [19].
- **Entropy Coding:** The hand-designed context models are replaced by CNN-based Bit Rate Estimation Nets to estimate the probability distribution of latent representations.

The most significant advantage of this approach is the capability for *joint optimization*. Unlike traditional codecs where modules are tuned individually, all components in the NVC framework—including motion estimation, compression, and residual coding—are implemented as neural networks and optimized simultaneously through a single loss function. This objective considers the trade-off between the number of compression bits and the reconstruction quality, allowing the modules to collaborate effectively to improve overall coding efficiency.

2.2.2 Temporal Prediction: Motion Vectors to Optical Flow

In traditional video coding standards like H.264 and H.265, temporal redundancy is reduced using block-based motion estimation. The frame is partitioned into rectangular blocks, and a search algorithm finds the best matching block in the reference frame. The displacement is represented by a Motion Vector (MV). While efficient, this approach assumes a rigid translational motion model, which often fails to capture complex non-rigid motions (e.g., deformation, rotation) and results in block artifacts at low bitrates.

Neural Video Coding addresses this limitation by adopting *Optical Flow* for motion estimation. Unlike the sparse motion vectors in traditional codecs, NVC estimates a dense motion field, providing a motion vector for every pixel.

Pyramid-based Motion Estimation

As illustrated in Figure 2.9(a), DVC utilizes a CNN-based optical flow network to estimate the motion field v_t. specifically employing the SPyNet (Spatial Pyramid Network) architecture [20]. SPyNet adopts a coarse-to-fine spatial pyramid structure, which allows it to efficiently handle large motions by estimating residual flow at multiple scales.

Crucially, in the DVC framework, this motion estimation module is not fixed but is **jointly optimized** end-to-end. Through Rate-Distortion Optimization (RDO), the network learns to generate optical flow maps that are not only accurate but also ”compressible.” As observed in, this joint training encourages the motion field to be sparse (similar to zero motion vectors in H.265) to minimize the bit cost effectively.

Motion Compensation and Refinement

In traditional codecs, motion compensation is a linear overlay operation. However, simple warping based on optical flow can introduce artifacts. To mitigate this, DVC introduces a dedicated **Motion Compensation Network** (post-warping). This CNN takes the warped frame, the reference frame, and the motion vectors as concatenated inputs to generate a refined predicted frame $\bar{x}_t$. This step effectively removes warping artifacts and serves a similar role to the loop filters in traditional standards but in a learnable, non-linear manner.

Compressing the Motion Information

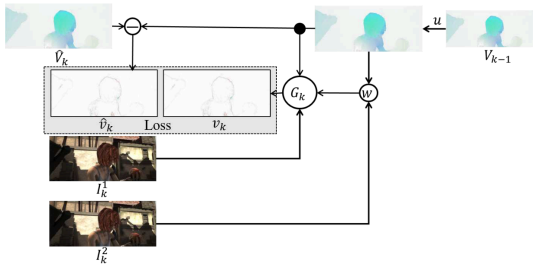
Since dense optical flow v_t contains significantly more data than block-based motion vectors, transmitting it directly is prohibitively expensive. DVC employs an *MV Encoder-Decoder Network* (Figure 2.9(b)) to compress this dense field.

- **MV Encoder:** Transforms the raw optical flow v_t into a compact latent representation m_t using a series of strided convolutions and non-linear Generalized Divisive Normalization (GDN) layers.

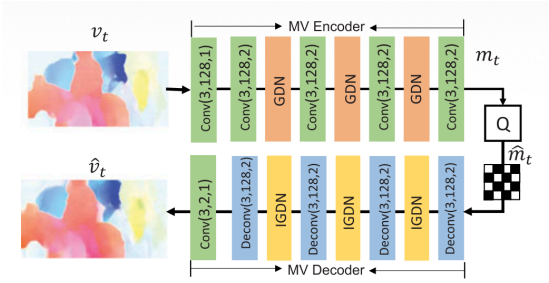
- **Quantization:** To enable end-to-end training (where standard rounding is non-differentiable), quantization is approximated by adding uniform noise during train-

ing, while standard rounding is used during inference.

- **MV Decoder:** Reconstructs the quantized latents $\hat{m}_t$ back into the motion field $\hat{v}_t$, which is then used for motion compensation.



(a) Optical Flow Estimation using a pyramid structure (SPyNet) to handle large motions [20].



(b) MV Encoder and Decoder Network designed to compress the dense optical flow [18].

Figure 2.9: The motion processing pipeline in DVC.

2.2.3 Spatial Transformation: Linear to Non-linear

In traditional frameworks, spatial redundancy is removed using linear transforms, notably the Discrete Cosine Transform (DCT) [21]. As shown in Fig. 2.10, DCT projects residuals onto fixed frequency basis functions, compacting energy into low-frequency coefficients for efficient quantization.

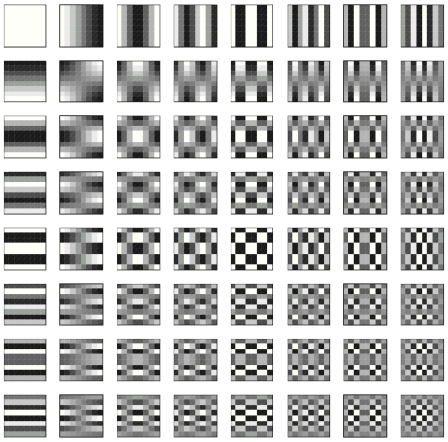


Figure 2.10: The 8×8 DCT basis functions used in traditional video coding standards.

Limitations of Linear Transforms

While effective, the block-based nature of DCT partitions images into independent 8×8 blocks, as seen in Fig. 2.11(b). This rigid partitioning often fails to capture structures crossing block boundaries, causing blocking artifacts at low bitrates and limiting rate allocation flexibility.

The NVC Approach: Learned Non-linear Transforms

NVC frameworks replace fixed bases with deep autoencoders (e.g., CNNs with GDN [22]) to map residuals into latent representations. Unlike the linear DCT, this learned transformation is highly non-linear and spatially continuous.

Fig. 2.11(c) visualizes the energy distribution of the learned latents. In contrast to disjointed DCT blocks, the NVC encoder acts as a semantic feature extractor, naturally concentrating activations on structural details like object boundaries. This **structural awareness** enables the Rate Control module to allocate bits based on visual saliency rather than fixed partitions, significantly improving perceptual quality.

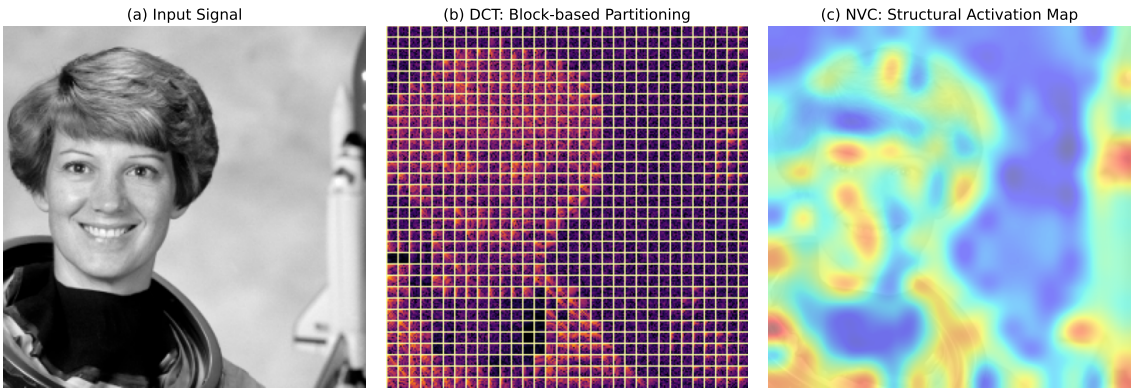


Figure 2.11: Visual comparison of spatial energy distribution. This energy heatmap is extracted from the latent space of the model [23] using the **CompressAI** library [24].

2.2.4 Differentiable Quantization and Optimization

While quantization is essential for compression, it poses a difficult problem for deep learning: the standard rounding operation, $Q(y) = \text{round}(y)$, is non-differentiable. Because its gradient is zero almost everywhere, it effectively cuts off the gradient flow needed for back-propagation.

Training Proxy: Uniform Noise Approximation

To make the system differentiable, NVC frameworks replace the rigid rounding operation with **additive uniform noise** during the *training phase*, a method proposed by Ballé et al. [25].

As shown in Figure 2.12 (Right), we approximate the discrete probability mass function of the quantized symbol $\hat{y}$ using the continuous density function of a noisy symbol $\tilde{y}$:

$$\tilde{y} = y + \eta, \quad \text{where } \eta \sim \mathcal{U}(-0.5, 0.5) \tag{2.3}$$

This **noise approximation** effectively simulates the error introduced by quantization ($\hat{y} - y$) while ensuring the gradients remain valid for optimization. Once training is complete, the model switches back to standard rounding ($\hat{y} = \text{round}(y)$) for the actual inference.

End-to-End Loss and Rate Estimation

Using this differentiable proxy allows us to train the entire compression system jointly. The Rate-Distortion Optimization (RDO) is directly integrated into the training loss:

$$\mathcal{L} = D(x, \hat{x}) + \lambda R(\tilde{y}) \tag{2.4}$$

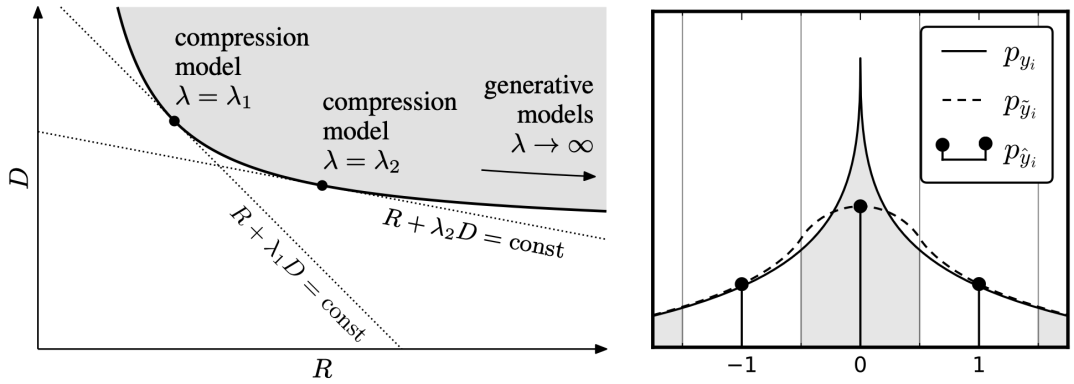


Figure 2.12: Quantization strategies in NVC. Left: The optimization landscape defined by λ. Right: To resolve the zero-gradient problem of hard quantization (dots), uniform noise (dashed line) is added during training as a differentiable proxy [25].

While calculating distortion D (e.g., MSE, MSSSIM) is straightforward, the rate term $R(\tilde{y})$ presents a challenge because counting the actual bits from an arithmetic coder is a non-differentiable process.

To address this, we rely on Shannon's source coding theorem[14], which states that the lower bound of the bit rate is the **Cross Entropy** between the marginal distribution of the latents and the learned probability model. Therefore, instead of counting bits, the loss function estimates the rate R as the **Negative Log-Likelihood (NLL)** of the noisy symbols:

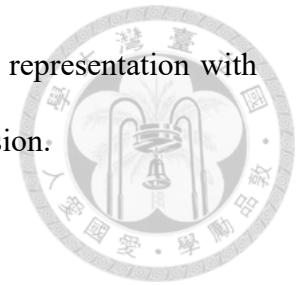
$$R(\tilde{y}) = \mathbb{E}_{\tilde{y} \sim p_{\tilde{y}}}[-\log_2 p_\theta(\tilde{y})] \tag{2.5}$$

where $p_\theta(\cdot)$ is the probability distribution predicted by the Entropy Model (see Section 2.2.5).

In our implementation, as the latent variables are convolved with a unit uniform distribution (due to the quantization noise), the probability of a symbol $\tilde{y}$ is calculated by integrating the learned distribution density (e.g., Laplace or Gaussian) over the corresponding quantization bin:

$$p_\theta(\tilde{y}) = \text{CDF}_\theta(\tilde{y} + 0.5) - \text{CDF}_\theta(\tilde{y} - 0.5) \tag{2.6}$$

Minimizing this NLL loss encourages the network to learn a latent representation with lower entropy, effectively reducing the bit rate required for transmission.



2.2.5 Entropy Coding: Context Models to Learned Priors

After quantization, the resulting discrete symbols must be compressed into a compact bitstream. While traditional standards employ fixed or adaptive lookup tables, NVC relies on Arithmetic Coding driven by learned probability distributions.

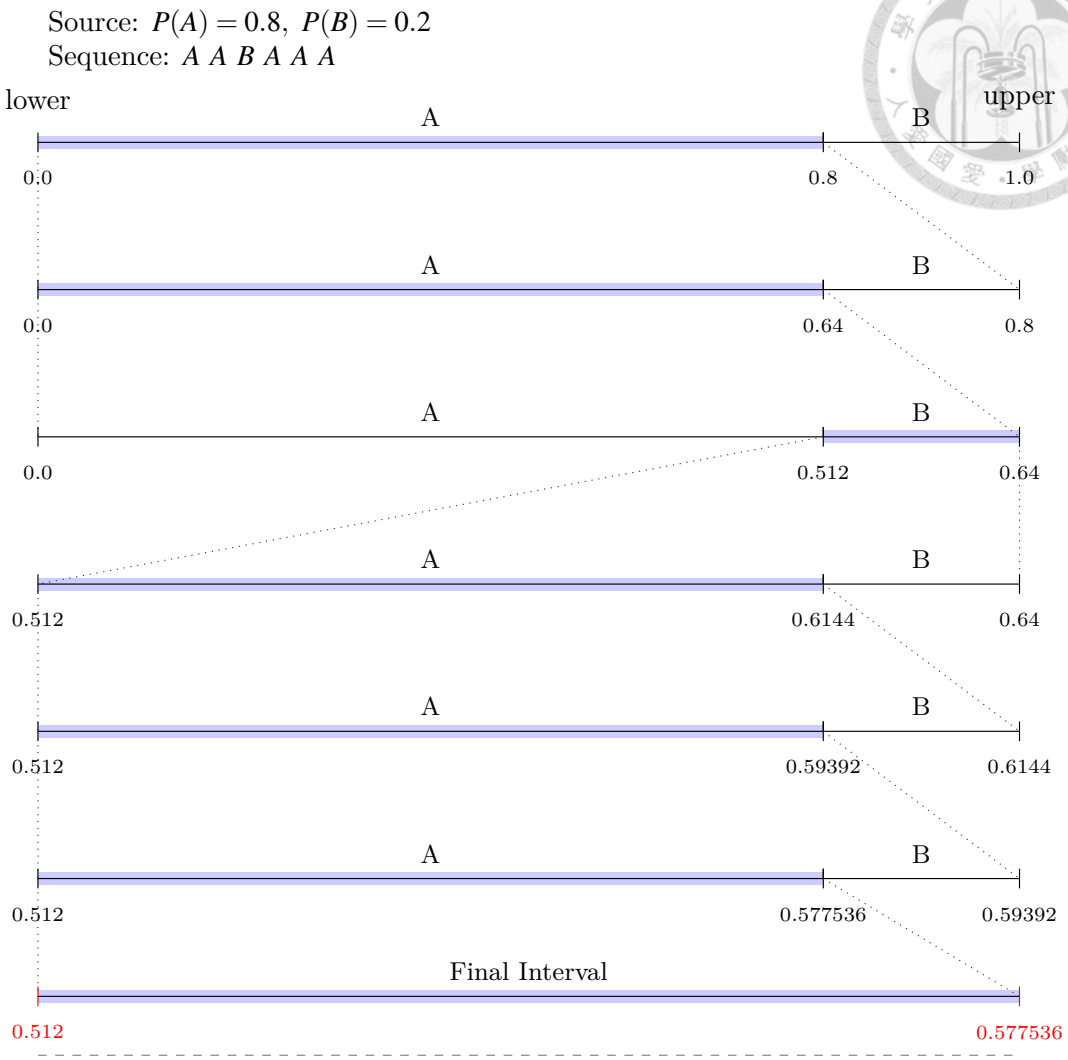
From Huffman to Arithmetic Coding

Recall from Section 2.1.2 that **Huffman coding** maps symbols to integer-length codewords. While optimal for symbol-by-symbol coding, this integer constraint becomes a significant bottleneck when handling symbols with very high probabilities (e.g., if $P(A) = 0.9$, the ideal code length is $-\log_2(0.9) \approx 0.15$ bits, but Huffman must assign at least 1 bit).

Arithmetic Coding (AC) overcomes this limitation by encoding a sequence of symbols into a single floating-point number, effectively allowing for the allocation of fractional bits. As illustrated in Figure 2.13, the probability interval $[0, 1)$ is recursively subdivided. A high-probability symbol retains a larger sub-interval, requiring fewer bits to resolve the final coordinate. This property allows AC to approach the theoretical **Entropy** limit (H) much more closely than Huffman coding.

Context Modeling and CABAC

The efficiency of Arithmetic Coding is strictly bounded by how accurately the estimated probability $P(\hat{y})$ matches the true data distribution. In H.264/AVC and H.265/



Final Interval: $[0.512, 0.577536)$
$0.512 \leq 17 \cdot 2^{-5} < 18 \cdot 2^{-5} \leq 0.577536$
Output Code(convert 17 to binary form): 10001 (5 bits)
Efficiency: $5/6 \approx 0.83$ bits/symbol

Figure 2.13: Illustration of Binary Arithmetic Coding.

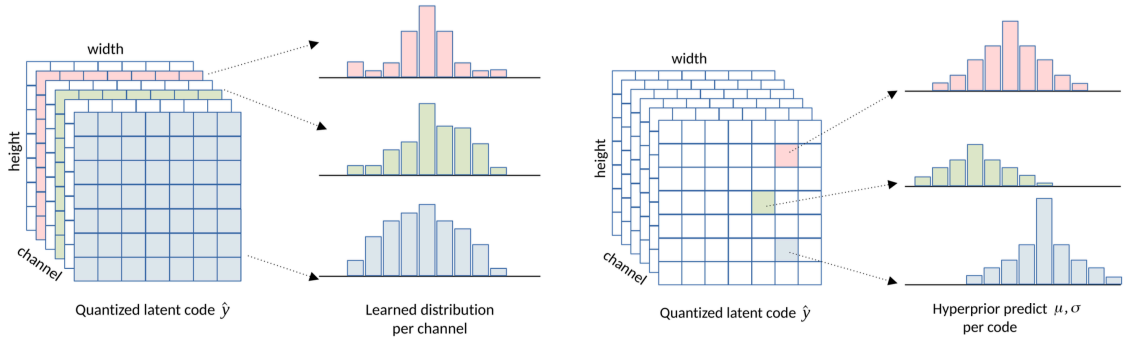
HEVC, this is handled by **CABAC (Context-based Adaptive Binary Arithmetic Coding)** [26]. CABAC utilizes hand-crafted context models that update on-the-fly based on previously coded symbols (neighbors). NVC adopts a similar philosophy but replaces these manual context updates with neural networks.

Learned Entropy Models

To provide precise probability estimates for the arithmetic coder, NVC employs CNN-based ”Entropy Models”.

Early approaches used a **Factorized Prior** (Figure 2.14(a)), assuming that each channel follows a fixed, independent distribution. However, this fails to capture the spatial variation of natural images—complex textures and smooth backgrounds often coexist in the same channel.

To address this, modern frameworks (e.g., DVC [18]) adopt the **Hyperprior** model (Figure 2.14(b)). This approach models the latent $\hat{y}$ as a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ where the parameters are spatially adaptive.



(a) Factorized Prior: Independent distributions. (b) Hyperprior: Spatially adaptive distributions.

Figure 2.14: Conceptual comparison of Entropy Models [27].

Figure 2.15 details the operational flow of the Hyperprior architecture. The network utilizes a hierarchical structure:

1. A **Hyper-Encoder** (h_a) summarizes the spatial information of the latent $\hat{y}$ into a side-information variable $\hat{z}$.

2. This $\hat{z}$ is transmitted first (as side info).

3. A **Hyper-Decoder** (h_s) uses $\hat{z}$ to estimate the mean and standard deviation scales (μ, σ) of the Gaussian distribution for each pixel in $\hat{y}$.

By conditioning the probability model on these parameters, the arithmetic coder can allocate more bits to complex regions (high σ) and fewer to smooth regions, significantly improving the rate-distortion performance.

Crucially, the parameters μ and σ predicted by the Hyper-Decoder are precisely what define the probability distribution p_θ used in the rate estimation loss (Eq. 2.5) discussed in Section 2.2.4. This end-to-end linkage ensures that the network minimizes the bitrate by learning to predict distributions that tightly match the actual statistics of the latent codes.

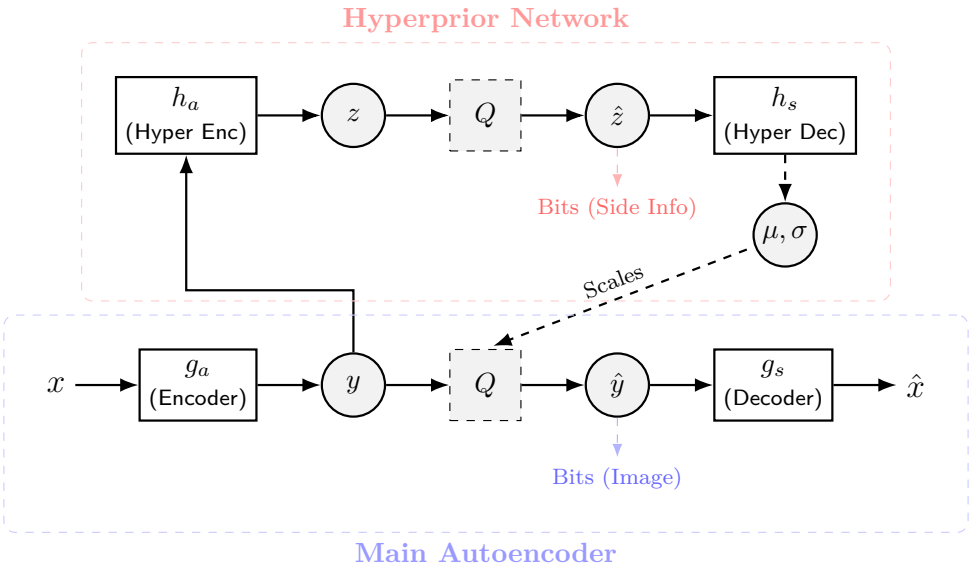


Figure 2.15: Operational diagram of the Hyperprior architecture. The input y is analyzed by the hyper-branch (top) to generate compressed side-information $\hat{z}$. The reconstructed $\hat{z}$ is then decoded to predict the mean and scales (μ, σ), which model the statistical distribution of $\hat{y}$ for efficient arithmetic coding.

Chapter 3

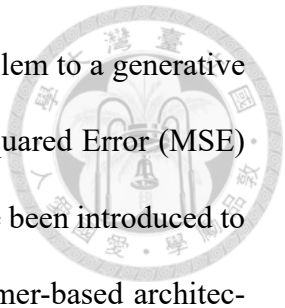
Literature Review

3.1 State-of-the-Art Neural Video Codecs

3.1.1 Developmental Progression

The foundation of Neural Video Coding (NVC) is fundamentally built upon advancements in Learned Image Compression (LIC). Ballé et al. [25] pioneered the use of Variational Autoencoders (VAEs) in this domain. To effectively capture spatial dependencies and match the statistics of natural images, Ballé further introduced the hyperprior model [23, 28] and Generalized Divisive Normalization (GDN) [22]. Unlike simple linear operations, GDN serves as a parametric non-linear transformation that effectively Gaussianizes the data density. By minimizing the mutual information between transformed components, GDN significantly enhances the efficiency of subsequent entropy coding, remaining a cornerstone in high-performance coding models [29].

Building on the success of LIC, Lu et al. proposed DVC (Deep Video Compression) [18, 30], the first end-to-end framework to successfully map the traditional hybrid coding pipeline—comprising motion estimation, motion compensation, and residual coding—onto neural networks. Early NVC was dominated by this ”Residual Coding” paradigm, utilizing Optical Flow techniques to capture temporal redundancy. To address occlusions and large displacements, Agustsson et al. introduced Scale-space flow [20, 31, 32], incorporating a blurring field to improve prediction robustness. This design has become foundational for mainstream architectures [33, 34] and subsequent video analysis tasks [24, 35].



Recently, NVC has begun shifting from a signal processing problem to a generative modeling problem [36]. To surpass the perceptual limits of Mean Squared Error (MSE) optimization, Generative Adversarial Networks (GANs) [37, 38] have been introduced to reconstruct realistic textures at low bitrates. Concurrently, Transformer-based architectures [5, 39, 40, 41, 42] are being explored to leverage attention mechanisms for global spatiotemporal feature interaction. However, finding the optimal balance between computational complexity and compression efficiency remains an ongoing challenge compared to mature CNN-based approaches.

Despite these advancements, a critical limitation of early methods was their reliance on pixel-domain residual coding. Subtracting predicted frames from current frames often leads to high-frequency information loss, particularly in high-texture or complex motion scenes. This bottleneck spurred the development of Conditional Coding—the core philosophy behind the DCVC series—which shifts operations into the feature domain to better preserve spatiotemporal correlations.

3.1.2 Conditional Coding and DCVC Architecture

From Residual to Conditional Coding

Traditional codecs typically adopt a ”residual coding” strategy, encoding the difference $r_t = x_t - \tilde{x}_t$. However, this linear subtraction often yields high-frequency, hard-to-compress signals and limits the modeling of non-linear spatiotemporal correlations. From an information-theoretic perspective, residual coding is merely a restricted special case of conditional coding. The entropy of residual coding is theoretically bounded by the entropy of conditional coding:

$$H(x_t - \tilde{x}_t) \geq H(x_t|\tilde{x}_t) \tag{3.1}$$

This inequality implies that relying solely on subtraction fails to fully exploit temporal correlations [43]. To break this theoretical bottleneck, Li et al. proposed the **Deep Contextual Video Compression (DCVC)** [44] framework. Instead of compressing pixel-domain differences, DCVC reformulates reconstruction as a conditional generation process:

$$\hat{x}_t = f_{dec}(\lfloor f_{enc}(x_t|\overline{x}_t)\rceil|\overline{x}_t) \tag{3.2}$$

where $\overline{x}_t$ is a high-dimensional context learned in the feature domain. As illustrated in Fig. 3.1, unlike standard predicted frames, this feature-rich context carries semantic information and texture patterns, enabling the entropy model to capture more accurate temporal priors.

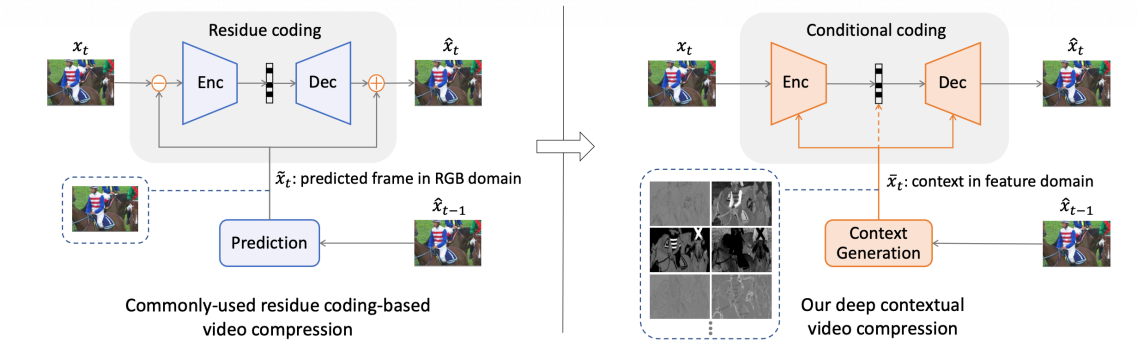


Figure 3.1: The paradigm shift from a residue coding-based framework to a conditional coding-based framework [44].

Temporal Context Mining (TCM)

While DCVC established the effectiveness of conditional coding, its initial implementation derived contexts from the previously decoded frame $\hat{x}_{t-1}$, which loses information during the projection back to the pixel domain. To extract more accurate temporal

information, **DCVC-TCM** [45] introduces the **Temporal Context Mining (TCM)** module (Fig. 3.2).

- **Feature Propagation:** Instead of using reconstructed frames, TCM propagates feature maps F_{t-1} directly. These features retain high-dimensional information that has not been degraded by RGB projection, ensuring better temporal coherence.

- **Multi-scale Context Mining:** Recognizing that motion and texture exhibit spatiotemporal non-uniformity, TCM employs a hierarchical structure to generate multi-scale contexts. The module performs feature warping and refinement across different resolutions to capture both large-scale motion and fine-grained details simultaneously.

- **Temporal Context Re-filling:** The mined contexts are injected into the encoder, decoder, and entropy model, ensuring that temporal information is utilized at every stage of the compression pipeline.

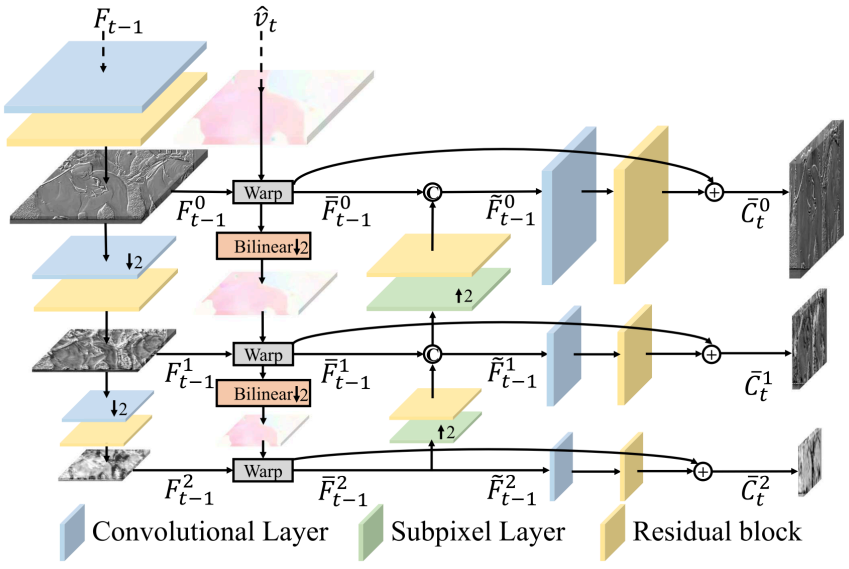


Figure 3.2: Architecture of the temporal context mining (TCM) module [45].

3.1.3 Single Model, Variable R-D Performance

In early NVC, a separate model had to be trained for each target bitrate by optimizing with a specific Lagrange multiplier (λ). This requirement led to high storage costs. Although transitional approaches such as internal feature modulation [46] and slimmable decoders [47] attempted to add adaptability, they failed to achieve continuous and smooth quality range within a single framework.

Adaptability via Quantization Scaling

Li et al. addressed this in **DCVC-HEM** [48] by introducing a quantization scaler (q_s) as a conditional input to the network. As shown in Fig. 3.3, the q_s interacts with the latent features in both the encoder and decoder. This mechanism allows the model to dynamically adjust the quantization step size and modify the feature distribution during inference. Consequently, a single trained model can support a wide range of bitrates without the need for retraining or storing multiple weights.

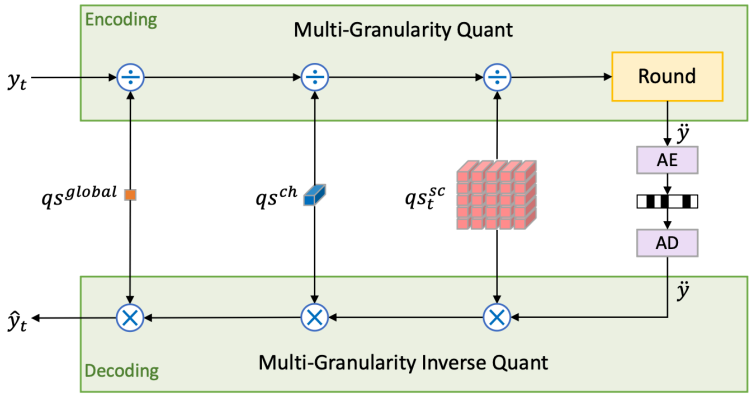


Figure 3.3: Multi-granularity quantization and the corresponding inverse quantization [48].

Quadtree Partition-Based Entropy Coding

To further enhance compression efficiency, recent research has focused on sophisti-

cated probabilistic modeling. While methods like hierarchical predictive learning [49] or multi-mode ensembles [50] have shown promise, **DCVC-DC** [51] pushes this boundary further by refining the probability estimation through a ”Diverse Contexts” architecture featuring a **Quadtree Partition** strategy.

As illustrated in Fig. 3.4, the latent representation $\hat{y}_t$ is split into channel groups and processed in four steps (Step 0 to Step 3).

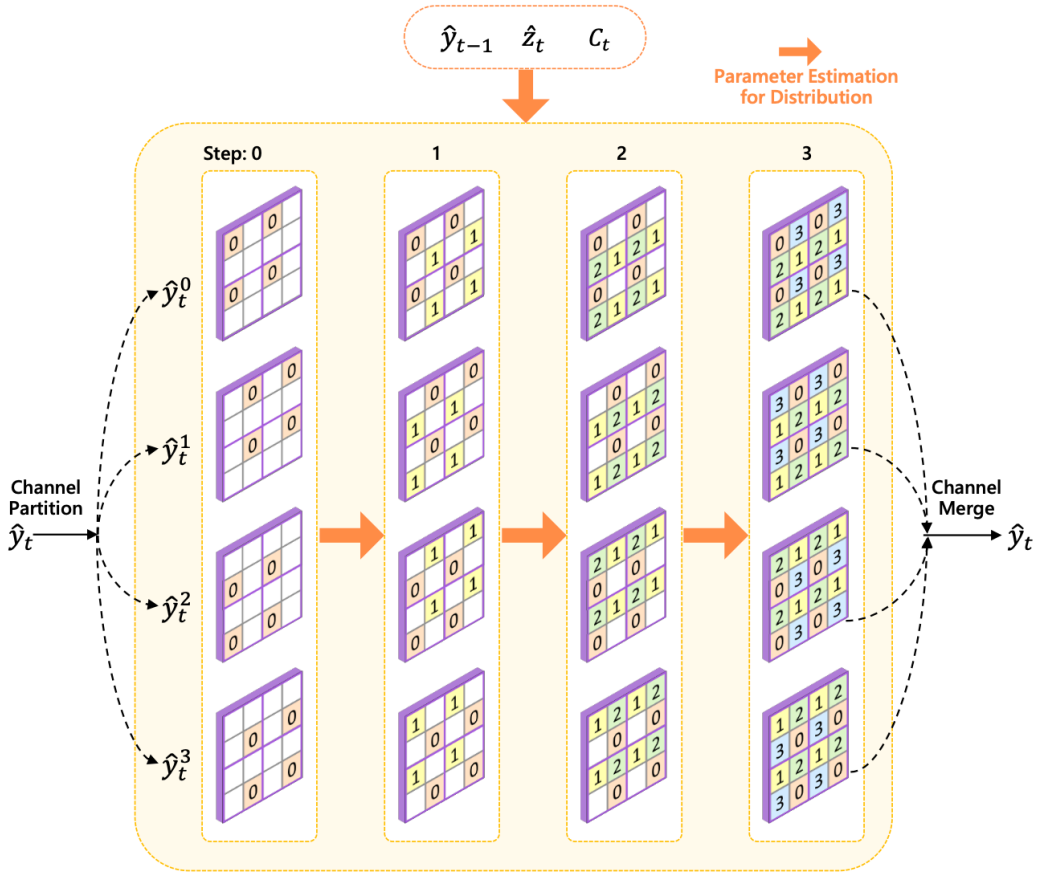


Figure 3.4: Entropy coding with quadtree partition. It fully mines the correlation from both spatial and channel dimensions [51].

1. **Step 0:** The anchor channels are coded independently without spatial neighbors.

2. **Steps 1-3:** Subsequent steps utilize the previously coded channels as contexts, employing 4, 4, and 8 neighbors, respectively.

This design allows the model to exploit both spatial neighbors (via the checkerboard-like pattern in the quadtree) and cross-channel correlations simultaneously. This approach significantly refines probability estimation compared to simple hyperpriors, setting a new standard for NVC entropy coding.

Solving the GOP Constraint: Hierarchical Quality Structure

To address severe error propagation in large Group of Pictures (GOP) [52], DCVC-DC introduces a **Hierarchical Quality Structure** inspired by H.266/VVC [4]. This strategy imposes a periodic quality pattern by assigning variable weights (w_t) to the distortion loss:

$$L = \frac{1}{T}\sum_{t}^{T}(w_t \cdot \lambda \cdot D + R) \tag{3.3}$$

Key frames receive larger weights (e.g., 1.2) to serve as high-quality references, while dependent frames are assigned smaller weights (e.g., 0.5). As shown in Fig. 3.5, this results in a "zig-zag" PSNR fluctuation similar to H.266. These periodic anchors effectively reset accumulated errors, stabilizing performance across long sequences and preventing the quality collapse common in earlier neural codecs.

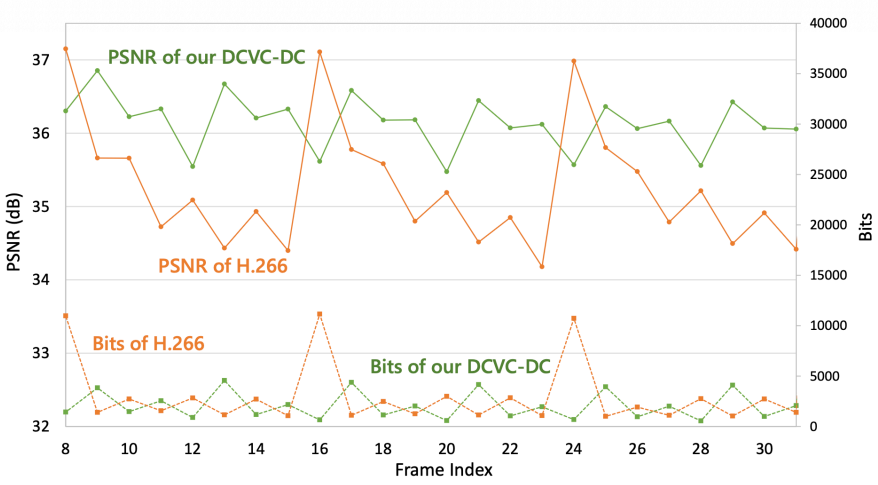


Figure 3.5: Hierarchical quality structure comparison [51].

3.1.4 Wider Quality Range and Real-Time Adaptation

Feature Modulation for Wide Range Coverage

While DCVC-DC enabled variable bitrates, performance often degraded at quality extremes. **DCVC-FM** [53] addresses this via a ”Feature Modulation” mechanism, applying learnable scaling factors in the feature domain to dynamically adjust the dynamic range based on target quality (Fig. 3.6). As shown in Fig. 3.7, this yields a wide 11.3 dB quality range, outperforming DCVC-DC and reducing bitrate by 30.2% over VTM-17.0. Crucially, this refined modulation ensures stability in long-sequence ”Single Intra-frame” coding, preventing the quality collapse often seen in earlier NVCs and offering robust support for potentially infinite video streams.

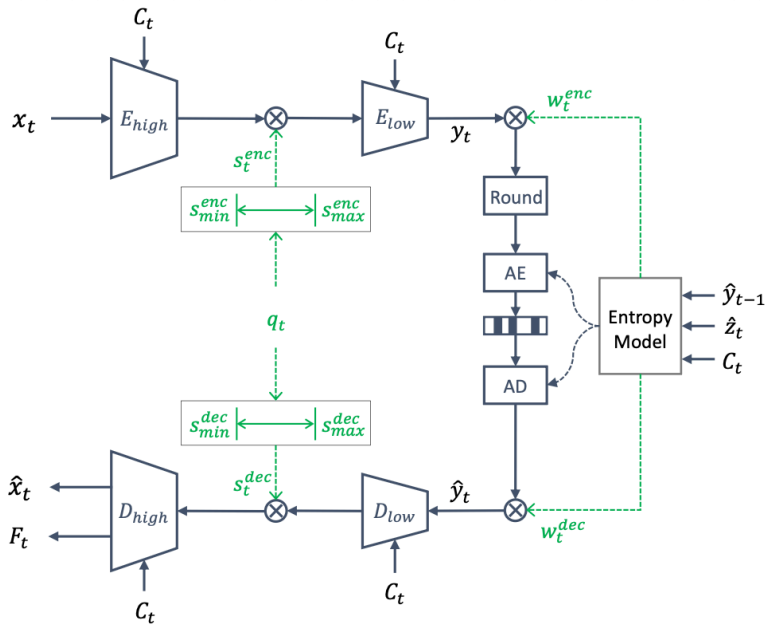


Figure 3.6: The Feature Modulation mechanism in DCVC-FM [53].

High-Throughput Neural Video Coding: DCVC-RT

While NVC offers superior compression, computational complexity has historically impeded real-time deployment. Previous acceleration efforts primarily focused on reduc-

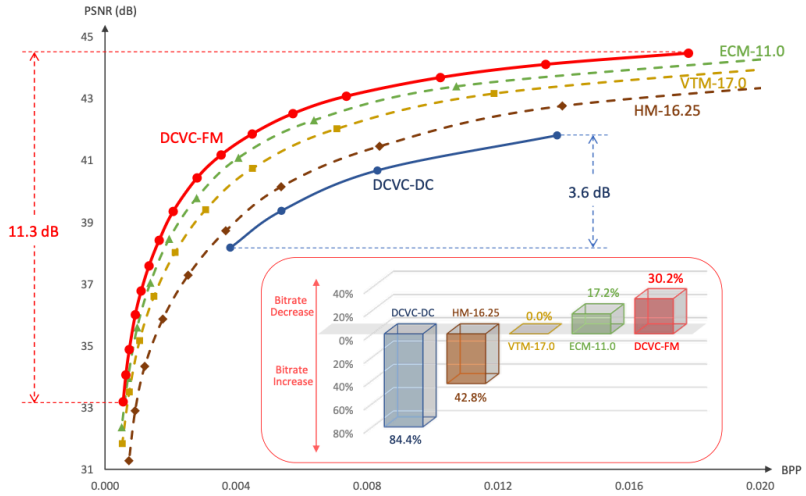


Figure 3.7: Rate-distortion performance comparison [53].

ing MACs [54, 55, 56], yet often **hit a speed limit** due to high "Operational Complexity"—specifically, the heavy overhead from memory access (I/O) and frequent function calls. To break this bottleneck, **DCVC-RT** [8] rethinks the coding paradigm by targeting system-level efficiency (Fig. 3.8):

- **Implicit Temporal Modeling:** Replaces expensive optical flow with a lightweight feature extractor to model correlations directly in the latent space (Fig. 3.8b), drastically reducing module calls.
- **Single-Scale Latent Representation:** Processes latents at a single low resolution (1/8) instead of a progressive pyramid. This minimizes high-resolution memory I/O while maintaining a sufficient receptive field.
- **Practicality Enhancements:** Incorporates model integerization (Int16) and module-bank-based rate control to ensure cross-device consistency and adaptability.

By addressing these operational bottlenecks, DCVC-RT achieves 1080p/30fps real-time performance on consumer GPUs with rate-distortion performance comparable to the sophisticated DCVC-FM.

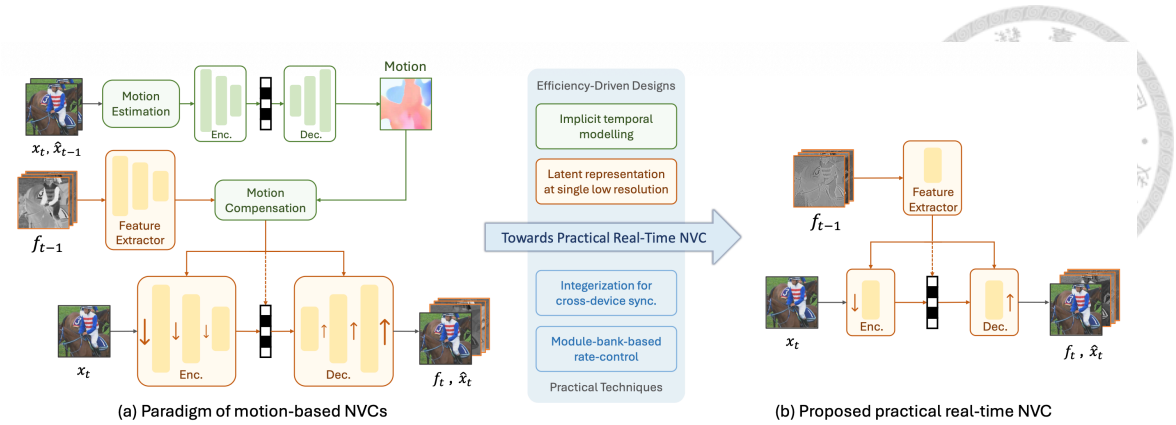


Figure 3.8: Comparison between (a) the traditional motion-based NVC paradigm and (b) the proposed DCVC-RT architecture. The shift to implicit modeling and single-scale latents significantly reduces operational complexity [8].

3.1.5 Summary

This section traced the evolutionary trajectory of Neural Video Coding (NVC), highlighting the decisive paradigm shift from pixel-domain residual coding to feature-domain conditional coding. Through the progressive development of the DCVC family—ranging from refined temporal context mining (DCVC-TCM) to versatile adaptability and real-time processing (DCVC-HEM, DCVC-RT)—NVC has successfully overcome early bottlenecks in accuracy and computational complexity. These advancements have elevated NVC from a theoretical exploration to a robust technology that consistently outperforms traditional standards like H.266/VVC in rate-distortion performance.

With the fundamental architecture now matured and superior compression efficiency established, the research focus naturally extends to system-level manageability. To bridge the gap between high-performance compression and practical video streaming, precise bitrate regulation becomes critical. This context underscores the value and necessity of developing dedicated rate control mechanisms for neural video codecs, which serves as the primary motivation for the subsequent discussion.

3.2 Rate Control Algorithms and Limitations

3.2.1 Rate Control in Traditional Codecs

Rate control aims to regulate the output bitstream to meet specific bandwidth constraints while minimizing coding distortion. This is formulated as a constrained optimization problem: minimize distortion D subject to a target rate R_c.

Limitations of Prior Models ($R-Q$ and ρ-Domain)

Early video coding standards (MPEG-2, H.264/AVC) predominantly utilized $R-Q$ models, such as the quadratic model [57], which assumed the Quantization Parameter (QP) was the sole determinant of bitrate. However, modern codecs like HEVC introduced significant dependencies between Rate-Distortion Optimization (RDO) and rate control, creating a "chicken and egg" dilemma [58]. While ρ-domain algorithms [59] proposed a linear relationship between rate and the percentage of zero coefficients, they rely on a one-to-one mapping between ρ and QP, which does not hold for the variable block-size transforms and flexible partitioning structure of HEVC.

The λ-Domain Rate Control Algorithm

To address the flexibility of HEVC, Li et al. proposed the λ-domain rate control algorithm [7], which has been adopted into the HEVC reference software (HM). This approach shifts the focus from QP to the Lagrange multiplier λ. The core is rooted in the unconstrained Lagrangian optimization problem:

$$J_{opt} = \arg\min_{\theta}(D + \lambda R) \tag{3.4}$$

where D is the distortion, R is the bitrate, and λ is the Lagrange multiplier. Geometrically,

λ represents the slope of the tangent line to the convex hull of the operational R-D curve, defined as $\lambda = -\frac{\partial D}{\partial R}$.

Unlike QP, which is discrete, λ is continuous and dictates the trade-off between rate and distortion. The authors discovered that for HEVC, the relationship between λ and bitrate (measured in bits per pixel, bpp) is accurately modeled by a hyperbolic function:

$$\lambda = \alpha \cdot bpp^{\beta} \tag{3.5}$$

where α and β are parameters related to the video source complexity. This model decouples rate estimation from the RDO process: the encoder first determines the optimal λ to satisfy the target bitrate using Eq. (3.5), and then performs RDO. Finally, the QP is derived from λ using the logarithmic relation $QP = 4.2005 \ln(\lambda) + 13.7122$ [60].

Hierarchical Bit Allocation Strategy

A distinct advantage of the high accuracy of the λ-domain model is its ability to support complex bit allocation strategies. The algorithm implements a unified three-level bit allocation scheme covering Group of Pictures (GOP), Picture, and Basic Unit (BU). To ensure smooth quality transitions and satisfy buffer constraints, the target bits for a GOP are first calculated using a sliding window (SW) mechanism rather than a rigid budget:

$$Target_{GOP} = \frac{R_{PicAvg} \times (N_{coded} + SW) - R_{coded}}{SW} \times N_{GOP} \tag{3.6}$$

where N_{coded} and R_{coded} represent the number of coded pictures and accumulated bits, respectively. Subsequently, these bits are distributed to individual pictures based on their hierarchical importance. Frames in lower temporal layers, which serve as critical references, are assigned higher weights (ω_{Pic}), while higher temporal layer frames receive

lower weights:

$$Target_{Pic} = \frac{Target_{GOP} - Coded_{GOP}}{\sum \omega_{Pic}} \times \omega_{CurrPic} \tag{3.7}$$

Finally, within each picture, the remaining bits are allocated to Basic Units (BUs) proportional to their spatial complexity, estimated via the squared Mean Absolute Difference (MAD^2) of the collocated region in the previous frame.

Model Parameter Update

To adapt to non-stationary video content, the parameters α and β in Eq. (3.5) are updated after encoding each unit using the Least Mean Square (LMS) method. The update rules minimize the error between the logarithmic real λ and the model-predicted λ:

$$\lambda_{comp} = \alpha_{old} bpp_{real}^{\beta_{old}} \tag{3.8}$$

$$\alpha_{new} = \alpha_{old} + \delta_\alpha \cdot (\ln \lambda_{real} - \ln \lambda_{comp}) \cdot \alpha_{old} \tag{3.9}$$

$$\beta_{new} = \beta_{old} + \delta_\beta \cdot (\ln \lambda_{real} - \ln \lambda_{comp}) \cdot \ln bpp_{real} \tag{3.10}$$

where δ_α and δ_β are fixed learning rates.

3.2.2 Rate Control Approaches in Neural Video Compression

Rate control for Learned Video Compression (LVC) is an emerging field. Unlike traditional codecs with standardized partitioning, NVCs often rely on end-to-end optimization, making traditional $R-Q$ models difficult to apply directly. Recent works have attempted to bridge this gap through model-based or fully neural approaches. Table 3.1 provides a theoretical comparison of the core mathematical models used in these studies,

contrasting the traditional approach with recent neural methods.

Table 3.1: Comparison of Rate Control Schemes: Models, Allocation Strategies, and Updating Mechanisms.

Study	RC Model	Bit Allocation Strategy	Updating Mechanism
[7]	R-λ Model $\lambda = \alpha \cdot R^{\beta}$	**Hierarchical (Sliding Window)** Uses a sliding window for GOP-level target, then cascades to pictures using fixed hierarchical weights.	**Iterative Update by Real Bits** Updates α, β iteratively. **Loss:** Minimizes error in $\ln \lambda$ domain: $(\ln \lambda_{real} - \ln \lambda_{comp})^2$.
[10]	R-λ Model $\lambda = \alpha \cdot R^{\beta}$ (Same as [7])	**Dependency-based** Derived from explicit inter-frame dependency modeling (p-dependency ratio).	**Iterative Update by Real Bits** Updates α, β and p. **Loss:** Minimizes error in $\ln R$ domain:$\frac{1}{2}(\ln R_{real} - \ln \hat{R})^2$.
[11]	R-Q_s Model $R = C \cdot Q_s^{-K}$	**Hierarchical** Directly adopts the sliding window and hierarchical weight mechanism from [7].	**Iterative Update by Real Bits** Progressive online updating of C, K. **Loss:** Minimizes error in $\ln R$ domain.
[12]	Neural Network $f(\text{feats}) \rightarrow \lambda$	**Neural Network** Extracts spatiotemporal features to adaptively allocate bitrates (MiniGoP $\rightarrow$ Frame).	**Pre-trained Inference** No iterative update. Relies on the generalization of the pre-trained Rate Implementation Network.

Inter-frame Dependency Modeling (Li et al., ICASSP 2022 [10])

To the best of our knowledge, Li et al. [10] proposed the first rate control scheme tailored for Learned Video Compression (LVC). This method is built upon the variable-rate framework by Lin et al. [46], enabling continuous bitrate adjustment via a Lagrange multiplier λ.

The core innovation is the **R-D-λ model**. In contrast to block-level dependency in conventional codecs, the authors model the frame-level dependency via a chain rule of partial derivatives. They introduce a scalar parameter p_t to quantify the linear dependency

between the distortion of consecutive frames:

$$p_t = \frac{\partial D_{t+1}}{\partial D_t}. \tag{3.11}$$

Physically, p_t represents the propagation of quality degradation; a higher value indicates that distortion in the reference frame strongly impacts the subsequent frame.

Staged Parameter Estimation

For rate implementation, the method adopts the hyperbolic model $\lambda = \alpha R^{\beta}$. To ensure robust estimation, a **staged update algorithm** is employed to decouple the R-D modeling from dependency modeling.

First, α and β are updated via gradient descent to minimize the log-squared rate error loss $l = \frac{1}{2}(\ln \frac{R_{real}}{\hat{R}_i})^2$. The update rules are derived as:

$$\begin{aligned} \alpha_i &= \alpha_{i-1} - \frac{k \cdot \alpha_{i-1}\beta_{i-1}}{2} \ln \frac{R_{real}}{\hat{R}_i}, \\ \beta_i &= \beta_{i-1} - \frac{k \cdot \beta_{i-1}}{2 \ln \hat{R}_i} \ln \frac{R_{real}}{\hat{R}_i}, \end{aligned} \tag{3.12}$$

where k is the step size set to 0.5. Subsequently, the dependency parameter p_i is updated in a delayed manner using the observed distortions:

$$p_i \approx \frac{D_i - f_i(R_i) - (D_{i-1} - f_{i-1}(R_{i-1}))}{D_{i-1} - D_{i-2}}, \tag{3.13}$$

where $f_i(R_i)$ represents the estimated distortion from the current R-D model. Notably, to prevent parameter instability during the update process, the authors implemented specific heuristic constraints: parameter changes were clamped to 10% for rate increases and 30% for rate decreases.

R-Q Model for NVC (Liao et al., ICASSP 2024 [11])

Addressing the limitations of applying fixed λ models to variable-rate networks, Liao et al. [11] proposed a Rate-Quality ($R - Q_s$) based control scheme. This approach is designed as a "plug-in" strategy for NVC frameworks where bitrate is regulated by a latent scaling parameter Q_s. As illustrated in Fig. 3.9, this scheme operates in a plug-in fashion, decoupling the rate estimation from the internal network architecture.

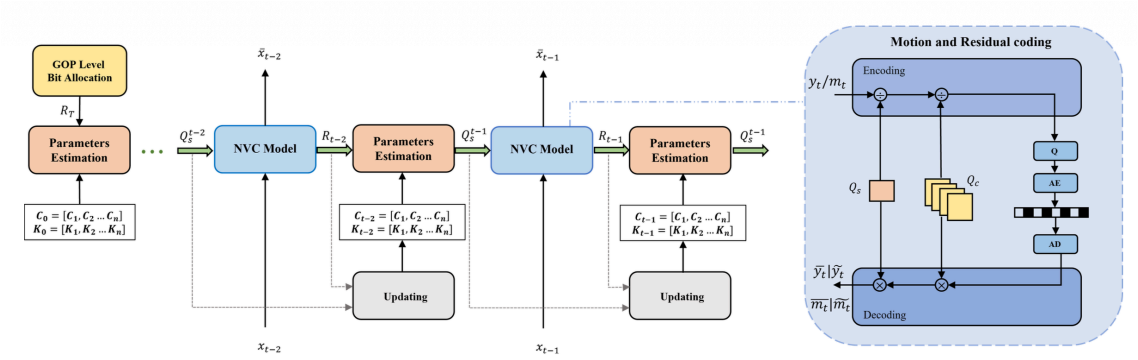


Figure 3.9: The framework of the Rate-Quality ($R - Q_s$) based rate control scheme proposed by [11].

R-Q Model and Parameter Update

Unlike the hyperbolic $R-\lambda$ relation, this method establishes a power-law relationship between the target bitrate R and the scaling factor Q_s, formulated as:

$$R = C \cdot Q_s^{-K}, \tag{3.14}$$

where C and K are content-dependent parameters. To adapt to non-stationary video content, C and K are updated frame-by-frame via gradient descent in the logarithmic domain. The update rules are defined as:

$$\begin{aligned} C_i' &= C_i - \eta \cdot (\ln \hat{R}_i - \ln R_{real}) \cdot C_i, \\ K_i' &= K_i - \mu \cdot (\ln \hat{R}_i - \ln R_{real}) \cdot \ln Q_{s,i}, \end{aligned} \tag{3.15}$$

where $\hat{R}_i$ is the predicted rate, R_{real} is the actual rate, and η, μ are step sizes.

Hierarchical Structure Preservation

A critical observation in this work is that traditional frame-level allocation often disrupts the hierarchical quality structure naturally learned by NVCs (where specific frames carry learned importance weights ω_i). By controlling the global scaling factor Q_s through the $R - Q_s$ model rather than forcing a rigid bit distribution derived from inter-frame dependency models, this method allows the network to implicitly allocate bits according to its internal weights, thereby minimizing R-D performance degradation.

Fully Neural Rate Control System (Zhang et al., ICLR 2024 [12])

While previous works adapted traditional empirical models, Zhang et al. [12] argue that such fixed mathematical forms may fail to capture the complex, non-linear rate-distortion characteristics of deep video compression. Consequently, they proposed the first fully neural network-based rate control system.

Two-Level Rate Allocation Strategy

To address the limitations of uniform allocation, the authors introduced a hierarchical strategy consisting of a **miniGoP-level** and a **Frame-level** allocation. First, the target bits for a miniGoP (R_{mg}) are determined using a sliding window approach. Subsequently, a **Weight Estimation Network** extracts spatiotemporal features from consecutive frames to assign an importance weight w_t to each frame. The target bitrate R_t for the current frame is calculated as:

$$R_t = \frac{R_{mg} - \hat{R}_{mg}}{\sum_{j=t}^{i+N_m-1} w_j} \times w_t. \tag{3.16}$$

This allows the system to implicitly model inter-frame dependency, allocating more bits to reference frames with higher impact features.

Rate Implementation via Feature Regression

This method employs a **Rate Implementation Network** to directly map the target bitrate R_t to the encoding parameter λ_t. The mapping is formulated as a regression problem conditioned on both current content and historical statistics. A critical normalization step is introduced to modulate the feature vector $\vec{V}_R$ derived from the input bitrate:

$$\vec{V}'_R = \frac{\vec{V}_R - \mu}{\theta}, \tag{3.17}$$

where (μ, θ) are normalization parameters fused from image and historical features. This design allows the network to predict precise λ values in a single forward pass, avoiding the latency of iterative search algorithms.

3.2.3 Comparative Analysis and Limitations

While the reviewed methods show progress, a systematic analysis reveals significant gaps when applying them to real-time scenarios. Table 3.2 details the experimental scope and specific limitations of each approach.

Inconsistency in Evaluation Protocols

A major hurdle in benchmarking NVC rate control is the lack of standardized evaluation protocols. As shown in Table 3.2, different studies adopt vastly different Group of Pictures (GOP) settings. For instance, Li et al. [10] evaluated their dependency model on short GOPs ($N = 12$), whereas Liao et al. [11] utilized longer sequences ($N = 192$). This

Table 3.2: Comparison of Experimental Settings and Practical Limitations.

Study	Experimental Context	Limitations
Li et al. [7]	**Codec**: Standard HEVC (HM 8.0) **Config**: Low Delay / Random Access **Settings**: GOP Size: 4; Frames: Not mentioned	Successful validation on traditional CODEC, but usually yields poor performance when applied to NVC.
Li et al. [10]	**Codec**: LVC (DVC base + Variable Rate) **Datasets**: HEVC, UVG **Settings**: GOP Size: 12; Frames: Not mentioned	Uses an extremely small GOP and is based on a very early (2019) NVC architecture.
Liao et al. [11]	**Codecs**: DVC, DMVC, DCVC-HEM, DCVC-DC **Datasets**: Vimeo-90K (Train), HEVC **Settings**: GOP Size: Not mentioned; Frames: 192	Many implementation details are unclear (e.g., Q_s and Q_c are shown, but reliance seems to be on Q_s only), and the GOP setting is completely missing.
Zhang et al. [12]	**Codecs**: DVC, FVC, DCVC, AlphaVC **Datasets**: Vimeo-90K (Train), HEVC, UVG, MCL-JCV **Settings**: GOP Size: 100; Frames: Not mentioned	Requires three-stage training, introduces computational overhead for real-time applications, and appears to have a training gap for latest CODECs with complex entropy models.

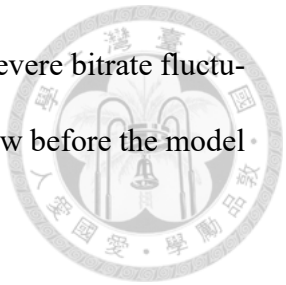
inconsistency often masks the potential stability issues of algorithms when exposed to the ”infinite GOP” (Intra-period = -1) setting commonly used in low-delay applications like cloud gaming, where error propagation can accumulate indefinitely.

Ambiguity in Parameter Initialization and Sensitivity

Most model-based approaches (e.g., $R - \lambda$ or $R - Q$ models) rely on iterative parameter updates. However, the literature often lacks transparency regarding **initialization strategies** and **hyperparameter sensitivity**.

- **Initialization:** The starting values of model parameters (α, β) significantly impact

the convergence speed. An improper initialization can lead to severe bitrate fluctuation in the first few GOPs, causing buffer overflow or underflow before the model stabilizes.



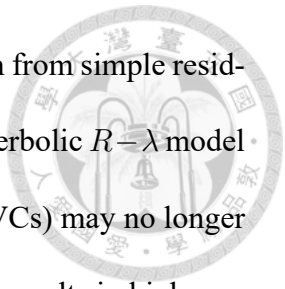
- **Learning Rates:** Algorithms utilizing gradient descent (e.g., Eqs. 9 and 12) are highly sensitive to the step size. Our preliminary experiments indicate that a fixed learning rate often fails to adapt to the drastic variance in scene complexity found in real-time scenarios. A rate too small causes sluggish adaptation during scene cuts, while a rate too large induces oscillation.

The Architecture-Control Mismatch with DCVC-RT

The most critical gap lies in the decoupling of rate control algorithms from the evolving NVC architectures, specifically regarding the **Conditional Coding** paradigm used in DCVC-RT.

1. **Obsolescence of Dependency Models:** Early methods like the p-domain model [10] relied on calculating inter-frame dependency chains ($\frac{\partial D_{t+1}}{\partial D_t}$). While effective for short GOPs or residual-based codecs (e.g., DVC), this approach becomes computationally intractable and numerically unstable for Conditional Coding. In DCVC-RT, temporal information is propagated via high-dimensional feature memories rather than simple pixel residuals, diluting the linear dependency assumption and making explicit dependency modeling inaccurate over long sequences.

2. **Model Mismatch in Feature Modulation:** Recent codecs like DCVC-FM and DCVC-RT achieve variable bitrates through Feature Modulation or Quantization Scaling (q_s). The underlying mathematical relationship between these latent scalers



and the output bitrate has shifted. As evidenced by the transition from simple residual coding to complex generative restoration, the traditional hyperbolic $R-\lambda$ model or the power-law $R-Q$ model (derived for HEVC or older NVCs) may no longer be the optimal fit. Blindly applying these models to DCVC-RT results in high prediction errors.

In conclusion, while DCVC-RT sets a new benchmark for compression efficiency, there is currently no rate control algorithm specifically optimized for its unique characteristics: **conditional coding, infinite P-frame structure, and feature-domain quantization**. This necessitates the development of a lightweight, adaptive, and mathematically grounded rate control scheme, which is the primary focus of this thesis.

Chapter 4

Proposed Method: Adaptive State-Space Design for Real-Time NVC Rate Control

4.1 System Overview

System Overview

In this study, we treat the Neural Video Compressor (NVC) system, specifically DCVC-RT, as a non-linear time-variant system. To focus on the derivation of the rate control algorithm, the encoder is abstracted as a Single-Input Single-Output (SISO) black-box model. As shown in Fig. 4.1, the system operates as a typical closed-loop control system.

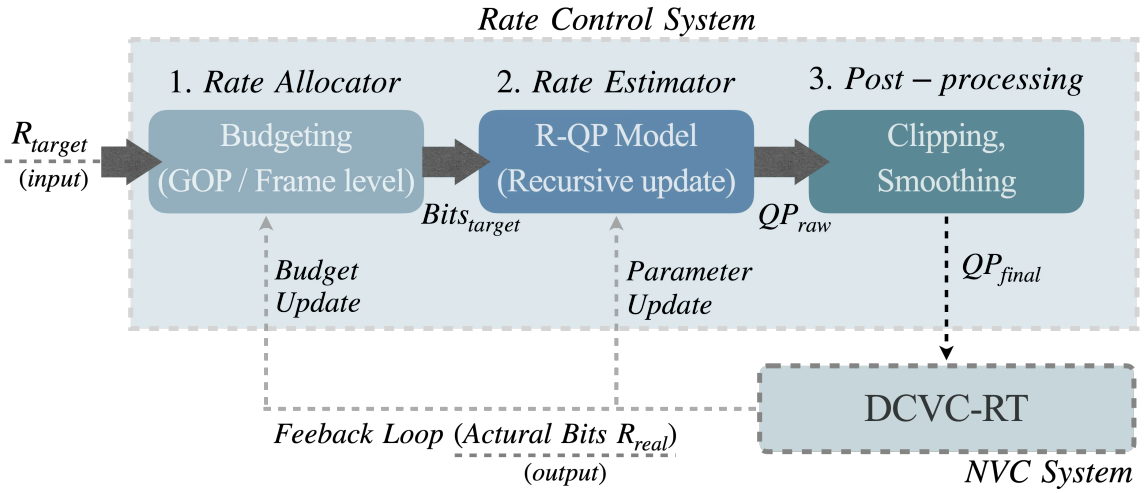
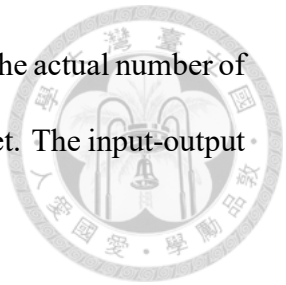


Figure 4.1: Overall architecture of the proposed rate control system. The DCVC-RT is treated as a black-box model. The controller determines the QP_{final} based on the target bitrate and updates its internal model using the feedback loop after encoding.

Let t denote the index of the video frame and X_t be the input raw image frame. For each frame, the objective of the Rate Control System is to determine a final Quantization



Parameter (QP_{final}) based on the input target bitrate R_{target}, such that the actual number of bits generated by the encoder, R_{real}, is as close as possible to the budget. The input-output relationship of the system can be expressed as:

$$R_{real,t}, \hat{X}_t = \mathcal{F}_{enc}(X_t, QP_{final,t} \mid \Theta_{net}) \tag{4.1}$$

where $\mathcal{F}_{enc}(\cdot)$ represents the encoding process of DCVC-RT, Θ_{net} denotes the fixed, pre-trained weights of the neural network, and $\hat{X}_t$ is the reconstructed image. Since we cannot modify Θ_{net} in real-time during the encoding process, QP becomes the sole control variable.

As illustrated in Fig. 4.1, our rate control system comprises three cascaded modules:

- **1. Rate Allocator:** Responsible for **Budgeting** at both the Group of Pictures (GOP) and Frame levels. It translates the global bitrate target R_{target} into a specific bit budget for the current frame, denoted as $Bits_{target}$.
- **2. Rate Estimator:** Maintains an internal **R-QP Model** to describe the relationship between quantization and bitrate. It takes the allocated $Bits_{target}$ as input and predicts the baseline quantization parameter, QP_{raw}. This module employs a recursive update mechanism to adapt to video content changes.
- **3. Post-processing:** Ensures the validity and stability of the control signal. It applies **Clipping** and **Smoothing** to the raw prediction QP_{raw} to generate the final control command QP_{final}, preventing buffer overflow and temporal flickering.

After the encoding of frame t is completed, the system utilizes a **Feedback Loop** to feed the actual generated bits R_{real} back to the controller. This feedback triggers the

Update of the budget status in the Rate Allocator and the **Parameter Update** in the Rate Estimator, thereby optimizing the decision-making for frame $t+1$.

4.2 R-Q Modeling and Empirical Analysis

Empirical R-Q Analysis via Exhaustive Search

To achieve precise rate control in a closed-loop system, the primary task is to establish a mathematical model between the Quantization Parameter (QP) and bits per pixel (R_{bpp}). Unlike traditional video coding standards (e.g., H.264/HEVC) which often model in the λ-domain, we aim to identify the characteristic that best represents the essence of the DCVC-RT architecture. To eliminate interference from prediction errors, we employed an **Exhaustive Search** strategy.

During the encoding of test sequences, we forced the encoder to traverse all allowed integer QP values ($QP \in [0, 63]$), thereby obtaining the Ground Truth R-Q Curve for each frame globally. Based on these high-density data points, we evaluated four common candidate models:

- Linear Model: $QP = a \cdot R_{bpp} + b$
- Power Model: $QP = a \cdot R_{bpp}^{b}$ (equivalent to $R = \alpha \cdot QP^{\beta}$)
- Exponential Model: $QP = a \cdot e^{b \cdot R_{bpp}}$
- Logarithmic Model: $QP = a \cdot \ln(R_{bpp}) + b$

Fig. 4.2 presents the curve fitting results using the exhaustive data. Experimental results show that the **Logarithmic Model** exhibits the highest goodness-of-fit (R^2 score) across test frames. The Linear model tends to underestimate QP in the low bitrate range,

while the Power model deviates in the high bitrate range. In contrast, the Log model accurately captures the characteristic of DCVC-RT where ”bitrate decays exponentially as QP increases.”

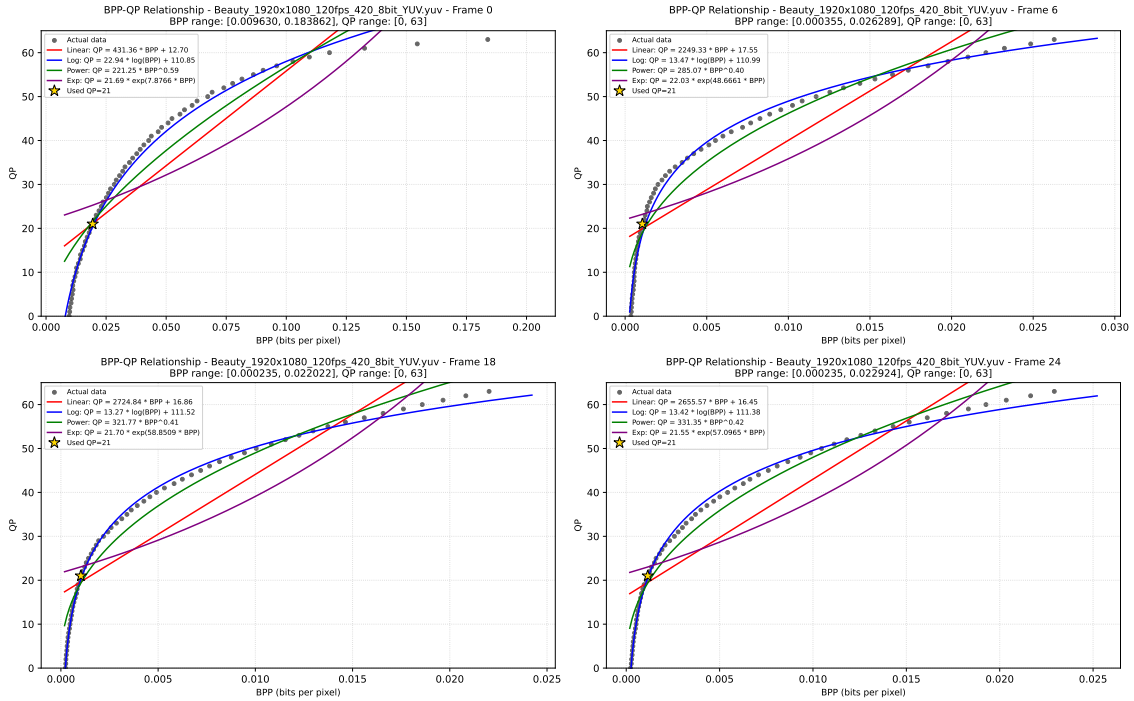


Figure 4.2: Empirical R-Q curve fitting based on exhaustive QP search data. The blue curve (Logarithmic model) demonstrates the highest fidelity to the actual data points across the entire bitrate range compared to other candidates.

Model Linearization and Objective Function

Based on the empirical evidence, we adopt the Log-linear R-Q Model:

$$QP_t = \alpha_t \cdot \ln(R_{bpp,t}) + \beta_t \tag{4.2}$$

This finding distinguishes our work from recent NVC rate control research which often utilizes Power Models (requiring non-linear optimization). Our model possesses a critical mathematical advantage: **Parameter Linearization**.

The equation can be viewed as a standard linear equation $y_t = \theta_t^T \phi_t$, where the target

variable y_t, parameter vector θ_t, and feature vector ϕ_t are defined as:

$$y_t = QP_t, \quad \theta_t = \begin{bmatrix} \alpha_t \\ \\ \beta_t \end{bmatrix}, \quad \phi_t = \begin{bmatrix} \ln(R_{bpp,t}) \\ \\ 1 \end{bmatrix} \tag{4.3}$$

We define the rate control problem as an optimization problem minimizing the squared prediction error. The instantaneous loss function is:

$$\mathcal{L}_t(\theta_t) = \left(QP_{real,t} - \theta_t^T \phi_t\right)^2 \tag{4.4}$$

This linear structure allows us to avoid complex non-linear solvers and instead employ efficient recursive algorithms for online estimation.

4.3 Adaptive Parameter Estimation via RLS

Recursive Least Squares (RLS) Algorithm

Due to the non-stationarity of video content, scene changes or fluctuations in complexity cause the optimal model parameters α_t and β_t to vary over time. To adapt to this time-variant characteristic, we leverage the **Recursive Least Squares (RLS)** algorithm.

The RLS algorithm is particularly suitable here because, as established in the previous section, our system model is linear in its parameters. RLS minimizes the cumulative weighted error:

$$\hat{\theta}_t = \arg\min_{\theta} \sum_{i=0}^{t} \lambda^{t-i} \mathcal{L}_i(\theta) \tag{4.5}$$

where $\lambda \in (0, 1]$ is the forgetting factor. This factor balances the importance of historical data against current data, enabling the model to adapt quickly to scene changes.

Fig. 4.3 illustrates the update mechanism. The specific steps for each frame t are as follows:

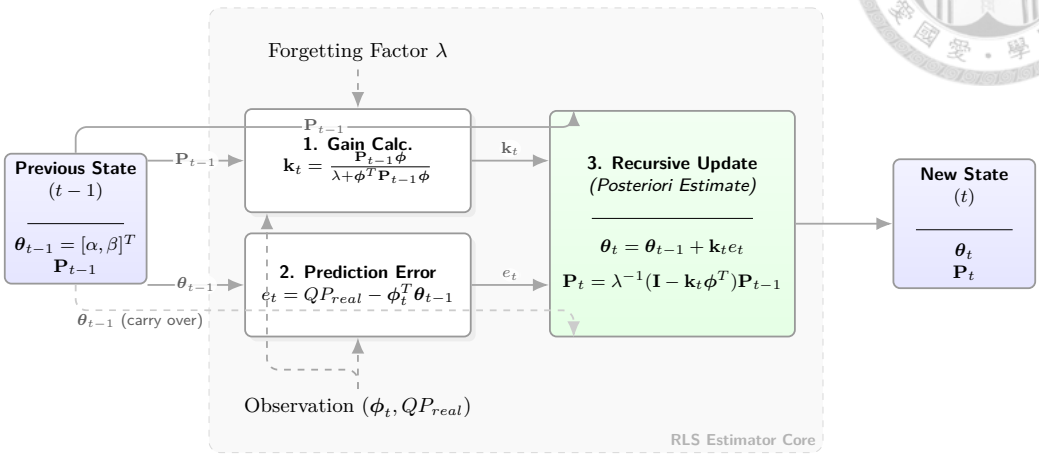


Figure 4.3: Internal mechanism of the RLS Estimator. The model updates its parameter state θ and covariance matrix $\mathbf{P}$ recursively based on the prediction error of the previous frame.

1. **Calculate Gain Vector:** Based on the previous covariance matrix $\mathbf{P}_{t-1}$ and current feature vector ϕ_t, compute the Kalman Gain $\mathbf{k}_t$:

$$\mathbf{k}_t = \frac{\mathbf{P}_{t-1}\phi_t}{\lambda + \phi_t^T\mathbf{P}_{t-1}\phi_t} \tag{4.6}$$

   In our implementation, we set $\lambda = 0.995$ for general P-frames to maintain stability while allowing adaptation.

2. **Calculate Prediction Error (Priori Error):** Compute the error e_t between the actually observed QP_{used} and the value predicted by the previous state:

$$e_t = QP_{used,t} - \phi_t^T\theta_{t-1} \tag{4.7}$$

3. **Update Model Parameters:** Correct the parameter estimate θ_t:

$$\theta_t = \theta_{t-1} + \mathbf{k}_t e_t \tag{4.8}$$

4. **Update Covariance Matrix:** Update the matrix **P** for the next iteration:

$$\mathbf{P}_t = \frac{1}{\lambda}(\mathbf{P}_{t-1} - \mathbf{k}_t \phi_t^T \mathbf{P}_{t-1}) \tag{4.9}$$

Through this recursive process, we obtain the optimal estimate $\theta_t = [\alpha_t, \beta_t]$ immediately after each frame is encoded.

Stability Analysis of Model Parameters

Now that the RLS estimator is defined, we must verify whether the chosen Log-linear model is conducive to the convergence of this algorithm. The *temporal stability* of model parameters is crucial for RLS; highly volatile parameters can lead to unstable Kalman gains and controller overshoot.

As illustrated in Fig. 4.4, we compared the parameter evolution of the Power Model ($QP = a \cdot R_{bpp}^b$) versus our Logarithmic Model over 600 frames. We selected two representative sequences: *Beauty* (Low SI/TI) and *ReadySteadyGo* (High TI).

The comparison reveals significant differences in parameter dynamics:

- **Power Model (Left):** The multiplier parameter a exhibits extreme variance and unpredictability. For *Beauty*, a fluctuates between 220 and 530; for *ReadySteadyGo*, it drifts between 120 and 240. Such a wide and erratic dynamic range places a heavy tracking burden on the RLS estimator, making it prone to divergence or noise sensitivity.

- **Log Model (Right):** In contrast, our model's slope parameter α (blue line) demonstrates remarkable stability. Regardless of sequence complexity, α remains consis-

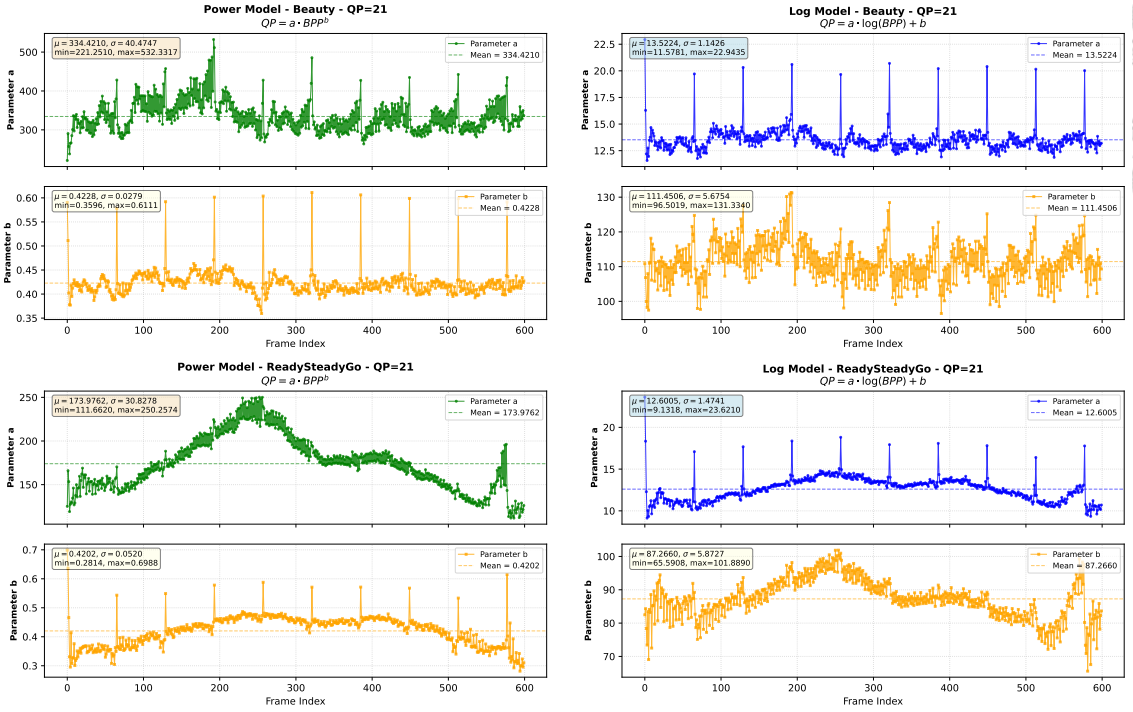


Figure 4.4: Comparison of parameter evolution between Power Model (Left) and Log Model (Right). Top row: *Beauty* (Low SI/TI); Bottom row: *ReadySteadyGo* (High TI). The Log Model parameters (α) exhibit stable periodicity within a predictable range ($10 \sim 20$), whereas the Power Model parameters fluctuate wildly ($100 \sim 550$) and lack clear patterns.

tently within the range of $[10, 20]$. It exhibits a stable baseline with regular spikes corresponding to the periodic Refresh Frames, displaying a clear stationary pattern.

This predictable behavior of α in the Log domain validates our choice of model, confirming it as a robust foundation for the proposed RLS-based control system.

4.4 Bitrate Allocation Strategy

Analysis and Limitations of Existing Schemes

Bit allocation serves as the decision-making core in closed-loop rate control. Its primary task is to translate the long-term target bitrate into specific bit budgets for individual frames.

The **Sliding Window-based allocation strategy**, originally established by Li et al. (IEEE TIP 2014) and recently adapted for Neural Video Coding by Liao et al. (ICASSP 2024), remains the industry standard for inter-frame bit regulation. This method relies on a fixed-size window to smooth traffic and has proven highly effective in traditional codecs (e.g., HEVC) and standard NVC approaches. Under typical configurations with short, fixed Group of Pictures (GOP) sizes (e.g., 16 or 32 frames), this strategy successfully balances budget fluctuations.

However, a critical misalignment arises when applying this standard strategy to the unique architecture of DCVC-RT. DCVC-RT operates on an ”Infinite P-frame” structure, inserting periodic ”Refresh Frames” (high-quality Intra-coded P-frames) to mitigate error propagation. These frames typically consume 3-5$\times$ the bits of normal frames.

Since the standard Sliding Window algorithm is not designed to anticipate such periodic, high-magnitude bursts within a continuous stream, it misinterprets the valid consumption of a Refresh Frame as a severe traffic overshoot. As illustrated in the bottom plot of Fig. 4.5, the allocator ”panics” upon encountering this spike (e.g., at Frame 64) and aggressively cuts the budget for subsequent frames. This results in a **post-refresh crash**—a sharp, unintended drop in quality immediately following the refresh—causing visible pumping artifacts.

Proposed Refresh-Aware Amortization Strategy

To adapt the robust Sliding Window logic to the DCVC-RT structure, we propose a **Refresh-Aware Amortization Strategy**. The core idea is to decouple the budget planning from the immediate consumption of refresh frames. We define two mechanisms:

Amortization (saving up bits) and *Virtual Feedback* (hiding the spike).

1. Amortization Scaling. We explicitly model the Refresh Period $L_{refresh}$. Assuming a Refresh Frame consumes k times the bits of a normal frame (empirically $k \approx 3.0 \sim 5.0$), we introduce a scaling factor ρ to levy a "bit tax" on normal frames:

$$\rho = \frac{L_{refresh}}{(L_{refresh} - 1) + k} \tag{4.10}$$

The baseline target T_{base} is adjusted to:

$$T_{base}(t) = \begin{cases} k \cdot \rho \cdot R_{target}, & \text{if } t \in \text{Refresh} \\ 1 \cdot \rho \cdot R_{target}, & \text{if } t \in \text{Normal} \end{cases} \tag{4.11}$$

This ensures that the extra cost of the refresh frame is amortized over the entire cycle, preventing budget depletion.

2. Virtual Feedback Mechanism. To prevent the underlying sliding window from reacting violently to the refresh spike, we normalize the feedback value. The allocator updates its history using a *Virtual Bitrate* R_{virt}:

$$R_{virt,t} = \begin{cases} R_{real,t}/k, & \text{if } t \in \text{Refresh} \\ R_{real,t}, & \text{if } t \in \text{Normal} \end{cases} \tag{4.12}$$

Fig. 4.5 validates the effectiveness of this design. In the **Bottom** plot (Baseline), a significant dip in bitrate is observed immediately after the peaks (e.g., Frame 64, 192), indicating that the controller is punishing the subsequent frames for the refresh cost. In contrast, the **Top** plot (Proposed) maintains a stable baseline. The red dashed box high-

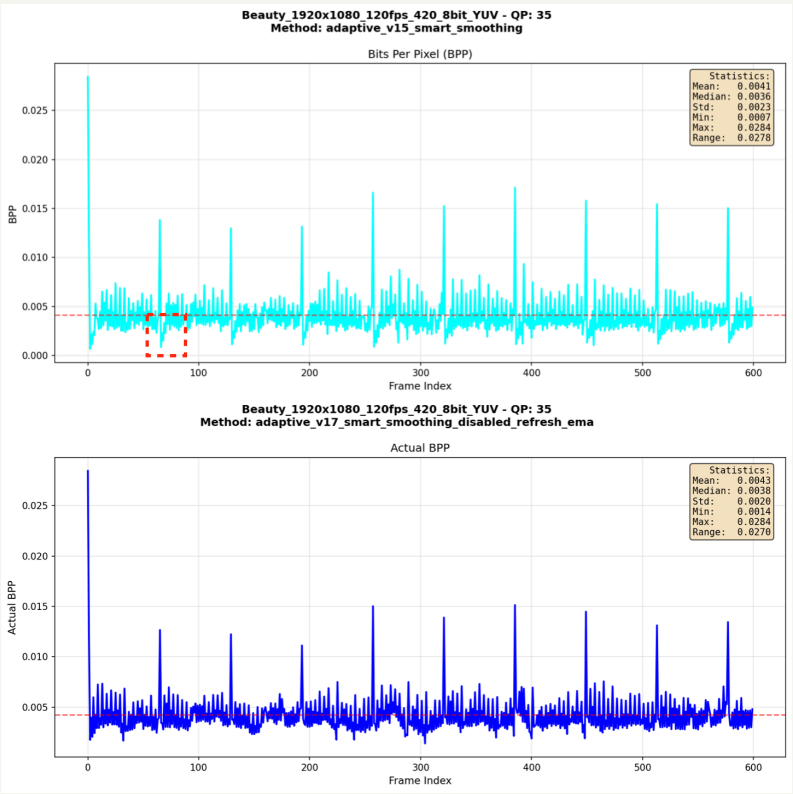


Figure 4.5: Comparison of bit allocation stability around Refresh Frames. **Top (Baseline):** The standard sliding window reacts to the refresh spike (Frame 64) by aggressively cutting the budget, causing a visible ”crash” (red box). **Bottom (Proposed):** Our strategy masks the spike, maintaining a stable baseline without post-refresh drops.

lights that even after a large refresh spike, the subsequent P-frames continue to receive a healthy bit budget, ensuring temporal consistency.

The complete allocation process is summarized in Algorithm 1.

4.5 Dynamic Smoothing and Estimation

The Stability-Accuracy Trade-off

Although the RLS estimator can capture changes in $R - QP$ model parameters in real-time, directly applying the raw predicted QP (QP_{raw}) to the encoder often leads to flickering artifacts. This is because the $R - Q$ relationship inherently contains noise, and neural encoders are extremely sensitive to minute QP changes. If we rely solely on instan-

Algorithm 1 Refresh-Aware Amortization Bit Allocation

Input: Target Bitrate R_{target}, Refresh Period $L_{refresh}$, Impact Factor k
Initialize: Rate Allocator $\mathcal{A}$ with sliding window size SW
Calculate Scaling Factor: $\rho \leftarrow L_{refresh}/((L_{refresh}-1)+k)$
for each frame t **do**
    Determine frame type: $is\_refresh \leftarrow (t \pmod{L_{refresh}} == 0)$
    if $is\_refresh$ **then**
        $T_{allocated} \leftarrow \mathcal{A}.allocate(k \cdot \rho \cdot R_{target})$
    else
        $T_{allocated} \leftarrow \mathcal{A}.allocate(1 \cdot \rho \cdot R_{target})$
    end if
    Encode frame t with budget $T_{allocated}$, get actual bits R_{real}
    Feedback Update:
    if $is\_refresh$ **then**
        $\mathcal{A}.update(R_{real}/k)$ ▷ Normalize to hide spike
    else
        $\mathcal{A}.update(R_{real})$
    end if
end for

taneous RLS predictions, QP may jump violently between adjacent frames. Conversely, using strong low-pass filtering (e.g., EMA with fixed coefficients) stabilizes the picture but makes the controller sluggish during scene cuts or traffic bursts, leading to buffer overflows or long-term bitrate drift. To break this ”zero-sum game” between stability and accuracy, we propose a **Smart Smoothing Controller**.

Smart Smoothing Controller

The core mechanism is the introduction of a **Safety Margin (ϵ)** as a decision threshold. The controller monitors the bitrate error rate E_t in real-time and dynamically switches between **Stable Mode** and **Emergency Mode**.

We define the error rate E_t as the average relative error within the sliding window:

$$E_t = \frac{|R_{real,t-1} - R_{target,t-1}|}{R_{target,t-1}} \tag{4.13}$$

The smoothing logic is defined as:

$$QP_{final,t} = \gamma_t \cdot QP_{final,t-1} + (1 - \gamma_t) \cdot QP_{raw,t} \tag{4.14}$$

where the smoothing coefficient γ_t is determined by the current error state:

$$\gamma_t = \begin{cases} 0.8, & \text{if } E_t \leq \epsilon \quad \text{(Stable Mode)} \\ 0.0, & \text{if } E_t > \epsilon \quad \text{(Emergency Mode)} \end{cases} \tag{4.15}$$

- **Stable Mode ($\gamma = 0.8$):** When the error is within the acceptable range ($\epsilon = 0.1$, i.e., 10% error), we adopt a strong smoothing strategy. This acts as a low-pass filter, removing high-frequency noise from the RLS prediction to ensure gradual QP changes and visual consistency.

- **Emergency Mode ($\gamma = 0.0$):** When the error exceeds the safety threshold (e.g., due to a scene cut causing model deviation), the system determines it is on the verge of losing control. The smoothing coefficient is immediately dropped to 0, bypassing the smoother and allowing the RLS prediction QP_{raw} to act directly on the encoder. This grants the system a high response speed, pulling the bitrate back to the target line within 1-2 frames.

4.6 Algorithm Summary

Combining the above sections, we summarize the proposed **Refresh-Aware Log-RLS Rate Control** algorithm in Algorithm 2. This algorithm integrates periodic amortization allocation, hybrid cycle RLS estimation, and smart smoothing control into a complete closed-loop system.

Algorithm 2 Proposed Refresh-Aware Rate Control Algorithm

Input: Target Bitrate R_{target}, Refresh Period $L_{refresh}$, Impact Factor k
Input: Video Sequence $\{X_t\}$, Forgetting Factor λ, Safety Margin ϵ
Initialize:
  RLS States $\Theta_{refresh}, \Theta_{normal}$ with $[\alpha_0, \beta_0]$
  Smoothing State $QP_{smooth} \leftarrow$ None
  Scaling Factor $\rho \leftarrow L_{refresh}/((L_{refresh} - 1) + k)$
for each frame $t = 1, 2, \ldots, N$ **do**
  // Step 1: Bitrate Allocation
  Determine frame type: $type \in \{REFRESH, NORMAL\}$
  if $type == REFRESH$ **then**
    $T_t \leftarrow$ Allocator.get_budget$(k \cdot \rho \cdot R_{target})$
    Select RLS Model: $\hat{\Theta}_t \leftarrow \Theta_{refresh}$
  else
    $T_t \leftarrow$ Allocator.get_budget$(1 \cdot \rho \cdot R_{target})$
    Select RLS Model: $\hat{\Theta}_t \leftarrow \Theta_{normal}$
  end if
  // Step 2: Model Prediction (RLS)
  Estimate raw QP: $QP_{raw} = \hat{\alpha} \cdot \ln(T_t) + \hat{\beta}$
  // Step 3: Smart Smoothing (Corrected)
  Calculate recent error ratio E_t
  if $E_t > \epsilon$ **then**
    $\gamma \leftarrow 0.0$ ▷ Emergency Mode: Fast Response
  else
    $\gamma \leftarrow 0.8$ ▷ Stable Mode: Smoothing
  end if
  if QP_{smooth} is None **then**
    $QP_{smooth} \leftarrow QP_{raw}$
  else
    $QP_{smooth} \leftarrow \gamma \cdot QP_{smooth} + (1 - \gamma) \cdot QP_{raw}$ ▷ Update Internal State
  end if
  $QP_{final} \leftarrow$ Clip(Round$(QP_{smooth}), QP_{min}, QP_{max})$
  // Step 4: Encoding & Update
  Encode frame X_t with QP_{final}, get actual bits R_{real}
  Update RLS:
    Compute error $e = QP_{final} - (\hat{\alpha} \ln(R_{real}) + \hat{\beta})$
    Update active state $\hat{\Theta}_t$ using RLS equations
  Update Allocator: ▷ Virtual Feedback Logic
  if $type == REFRESH$ **then**
    Allocator.update(R_{real}/k)
  else
    Allocator.update(R_{real})
  end if
end for

Chapter 5

Experiments and Results

5.1 Experimental Setup

To comprehensively evaluate the effectiveness of the proposed rate control scheme, we conducted extensive experiments on standard video compression benchmarks. This section details the datasets used, the implementation specifics of our framework and baselines, and the evaluation metrics employed.

5.1.1 Datasets

We selected three widely used datasets to evaluate the performance of our method across diverse video contents and motion characteristics. To ensure a fair and consistent comparison, all video sequences were standardized to a resolution of 1920×1080. Furthermore, we adopted a ”Low-Delay P” configuration (intra period set to -1), where only the first frame is encoded as an I-frame, and all subsequent frames are encoded as P-frames. We utilized the full length of each video sequence without cropping to test the long-term stability of the rate control algorithm.

- **UVG Dataset** [61]: This dataset contains high frame rate (120fps) 4K sequences. For our experiments, we downsampled these sequences to 1080p. The UVG dataset is characterized by high temporal redundancy and smooth motion, making it suitable for evaluating the efficiency of inter-frame prediction and rate stability over long durations.

- **MCL-JCV Dataset** [62]: The MCL-JCV dataset comprises 30 video sequences with a resolution of 1920×1080. It covers a wide variety of content types, including distinct textures and motion patterns (e.g., camera panning, zooming, and complex object motion). This diversity is crucial for assessing the generalization capability of the rate control model.

- **HEVC Class B Dataset** [3]: We utilized the standard Class B sequences from the HEVC common test conditions. These sequences are natively 1920×1080 and are widely used in the literature [28, 51] for benchmarking video compression performance. They present challenging scenarios with varying degrees of object motion and background complexity.

5.1.2 Implementation Details

5.1.2.1 Baseline Methods

We compared our proposed method against several state-of-the-art baselines to validate its performance. The baselines include:

- **Anchor (DVC/FVC without RC):** The original learned video compression model running with fixed λ values, serving as the upper bound for R-D performance.

- **Baseline A [Li et al., ICASSP 2022]:** A representative method utilizing an R-D-λ model with iterative parameter updates.

- **Baseline B [Zhang et al., ICLR 2024]:** A neural rate control approach using a two-level allocation strategy.

- **[Your Baseline Name]:** [Description of any other specific baseline you imple-

mented].

5.1.2.2 Parameter Settings

Our proposed framework was implemented using PyTorch on an NVIDIA [INSERT GPU MODEL, e.g., RTX 3090] GPU.

For the **Rate-Quality (R-Q) Model**, we utilized the [INSERT MODEL TYPE, e.g., Logarithmic] relationship as defined in Equation [X]. The initial parameters were set to $\alpha = [VALUE]$ and $\beta = [VALUE]$. The **neural network components** (e.g., the bit allocation network) were optimized using the Adam optimizer with an initial learning rate of 1×10^{-4}. The learning rate was decayed by a factor of 0.1 every [INSERT EPOCHS] epochs.

During inference, the target bitrate R_{target} was set based on the average bitrate of the anchor method at four quality levels ($\lambda \in \{256, 512, 1024, 2048\}$ or corresponding Q-levels). The sliding window size for the bitrate buffer was set to [INSERT VALUE, e.g., 40 frames].

5.1.3 Evaluation Benchmarks

To provide a holistic evaluation, we employed four key metrics focusing on accuracy, quality, stability, and efficiency.

5.1.3.1 Rate Control Accuracy

The primary goal of rate control is to minimize the deviation between the target bitrate and the actual output bitrate. We measure this using the absolute Bitrate Error (BRE), defined as:

$$\Delta R = \left| \frac{R_{target} - R_{actual}}{R_{target}} \right| \times 100\% \tag{5.1}$$

where R_{target} is the allocated bit budget and R_{actual} is the actual file size of the encoded sequence.

5.1.3.2 R-D Performance

We evaluate the coding efficiency using the Bjontegaard Delta Rate (BD-Rate) [63]. BD-Rate measures the percentage of bitrate savings required to achieve the same objective quality compared to an anchor. We utilize two quality metrics for this calculation:

- **PSNR (Peak Signal-to-Noise Ratio):** Evaluating pixel-level fidelity.

- **MS-SSIM (Multi-Scale Structural Similarity):** Evaluating perceptual quality.

5.1.3.3 Bitrate Stability

To assess the smoothness of the output bitstream, which is critical for streaming applications, we analyze the frame-level bitrate fluctuation. Lower variance in frame-level bitrate indicates a more stable rate control mechanism.

5.1.3.4 Time Overhead

We measure the computational complexity using the Time Ratio (TR), defined as the total encoding time of the method with rate control divided by the encoding time of the anchor (without rate control):

$$TR = \frac{T_{RC}}{T_{Anchor}} \tag{5.2}$$

A TR close to 1.0 indicates negligible computational overhead.

5.2 Performance Comparison

Table 5.1 summarizes the quantitative results on the HEVC Class B, UVG, and MCL-JCV datasets. The results are averaged across all sequences within each dataset.

Table 5.1: Comparison of Rate Error (ΔR), BD-Rate (PSNR), and Time Ratio (TR) on three benchmark datasets. The anchor is the DVC model without rate control.

Dataset	Method	ΔR (%) ↓	BD-Rate (%) ↓	TR ↓
HEVC Class B	Baseline A [51]	5.04	20.21	1.01
	Baseline B [28]	1.35	-10.99	1.04
	Ours	**0.45**	**-11.50**	**1.02**
UVG	Baseline A [51]	6.24	16.69	1.00
	Baseline B [28]	2.82	-11.61	1.05
	Ours	**0.88**	**-12.10**	**1.01**
MCL-JCV	Baseline A [51]	4.50	15.30	1.02
	Baseline B [28]	2.79	-8.78	1.06
	Ours	**1.10**	**-9.20**	**1.03**

Rate Control Accuracy As shown in Table 5.1, our method achieves the lowest bitrate error across all datasets. Specifically, on the challenging HEVC Class B dataset, our method maintains a ΔR of only **0.45%**, whereas Baseline A and B exhibit errors of 5.04% and 1.35%, respectively. This demonstrates the robustness of our [INSERT

YOUR MECHANISM, e.g., parameter updating strategy] in adhering to the target bitrate constraint.

R-D Performance Regarding coding efficiency, our method demonstrates superior performance. While traditional rate control schemes often degrade R-D performance (positive BD-Rate) due to suboptimal bit allocation, our method achieves a BD-Rate saving of **-11.50%** on HEVC Class B. This indicates that our rate allocation strategy not only meets the rate constraints but also intelligently distributes bits to optimize reconstruction quality, outperforming even the variable-rate baselines.

Computational Efficiency The Time Ratio (TR) column highlights the efficiency of our approach. With an average TR of approximately **1.02**, our method introduces only a marginal 2% increase in encoding time compared to the anchor. This confirms that our rate control module is lightweight and suitable for practical applications, contrasting with optimization-based approaches that may require iterative pre-encoding.

5.3 Ablation Studies and Parametric Analysis

Chapter 6

Conclusion

6.1 Summary of Contributions

6.2 Limitations and Future Work

References

[1] Ericsson, "Ericsson Mobility Report," Ericsson, Stockholm, Sweden, Tech. Rep., Jun. 2025, June 2025 edition. Accessed: Nov. 9, 2025. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report.

[2] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h. 264/avc video coding standard," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 560–576, 2003.

[3] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[4] B. Bross, Y.-K. Wang, Y. Ye, S. Liu, J. Chen, G. J. Sullivan, and J.-R. Ohm, "Overview of the versatile video coding (vvc) standard and its applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3736–3764, 2021.

[5] D. Liu, Y. Li, J. Lin, H. Li, and F. Wu, "Deep learning-based video coding: A review and a case study," *ACM Computing Surveys (CSUR)*, vol. 53, no. 1, pp. 1–35, 2020.

[6] J. Ascenso, E. Alshina, and T. Ebrahimi, "The jpeg ai standard: Providing efficient human and machine visual data consumption," *Ieee Multimedia*, vol. 30, no. 1, pp. 100–111, 2023.

[7] B. Li, H. Li, L. Li, and J. Zhang, "λ domain rate control algorithm for High Efficiency Video Coding," *IEEE Transactions on Image Processing*, vol. 23, no. 9, pp. 3841–3854, 2014.

[8] Z. Jia, B. Li, J. Li, W. Xie, L. Qi, H. Li, and Y. Lu, "Towards practical real-time neural video compression," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 12 543–12 552.

[9] *Enhanced compression model (ecm) reference software*, https://vcgit.hhi.fraunhofer.de/ecm/, JVET experimental reference software.

[10] Y. Li, X. Chen, J. Li, J. Wen, Y. Han, S. Liu, and X. Xu, "Rate control for learned video compression," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2022, pp. 2829–2833.

[11] S. Liao, C. Jia, H. Fan, J. Yan, and S. Ma, "Rate-quality based rate control model for neural video compression," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2024, pp. 4215–4219.

[12] Y. Zhang, G. Lu, Y. Chen, S. Wang, Y. Shi, J. Wang, and L. Song, "Neural rate control for learned video compression," in *The Twelfth International Conference on Learning Representations*, 2024.

[13] S. S. Haykin, *Adaptive filter theory*. Pearson Education India, 2002.

[14] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[15] K. Sayood, *Introduction to data compression*. Morgan Kaufmann, 2017.

[16] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video enhancement with task-oriented flow," *International Journal of Computer Vision*, vol. 127, no. 8, pp. 1106–1125, 2019.

[17] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 2007.

[18] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao, "Dvc: An end-to-end deep video compression framework," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 11 006–11 015.

[19] K. O'shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[20] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4161–4170.

[21] T. Wiegand, F. Jeltsch, I. Hanski, and V. Grimm, "Using pattern-oriented modeling for revealing hidden information: A key for reconciling ecological theory and application," *Oikos*, vol. 100, no. 2, pp. 209–222, 2003.

[22] J. Ballé, V. Laparra, and E. P. Simoncelli, "Density modeling of images using a generalized normalization transformation," *arXiv preprint arXiv:1511.06281*, 2015.

[23] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," *arXiv preprint arXiv:1802.01436*, 2018.

[24] J. Bégaint, F. Racapé, S. Feltman, and A. Pushparaja, "Compressai: A pytorch library and evaluation platform for end-to-end compression research," *arXiv preprint arXiv:2011.03029*, 2020.

[25] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," *arXiv preprint arXiv:1611.01704*, 2016.

[26] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the h.264/avc video compression standard," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 620–636, 2003.

[27] C.-H. Huang and J.-L. Wu, "Unveiling the future of human and machine coding: A survey of end-to-end learned image compression," *Entropy*, vol. 26, no. 5, p. 357, 2024.

[28] D. Minnen, J. Ballé, and G. D. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," *Advances in neural information processing systems*, vol. 31, 2018.

[29] G.-H. Wang, J. Li, B. Li, and Y. Lu, "Evc: Towards real-time neural image compression with mask decay," *arXiv preprint arXiv:2302.05071*, 2023.

[30] G. Lu, X. Zhang, W. Ouyang, L. Chen, Z. Gao, and D. Xu, "An end-to-end learning framework for video compression," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 10, pp. 3292–3308, 2020.

[31] E. Agustsson, D. Minnen, N. Johnston, J. Balle, S. J. Hwang, and G. Toderici, "Scale-space flow for end-to-end optimized video compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8503–8512.

[32] Y. Wang, L. Lipson, and J. Deng, "Sea-raft: Simple, efficient, accurate raft for optical flow," in *European Conference on Computer Vision*, Springer, 2024, pp. 36–54.

[33] J. Yang, C. Yang, Y. Zhai, Q. Wang, X. Pan, and R. Wang, "Improving learned video compression by exploring spatial redundancy," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2024, pp. 2860–2864.

[34] Y. Shi, Y. Ge, J. Wang, and J. Mao, "Alphavc: High-performance and efficient learned video compression," in *European Conference on Computer Vision*, Springer, 2022, pp. 616–631.

[35] C. Reich, B. Debnath, D. Patel, T. Prangemeier, D. Cremers, and S. Chakradhar, "Deep video codec control for vision models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 5732–5741.

[36] Z. Chen, H. Sun, L. Zhang, and F. Zhang, "Survey on visual signal coding and processing with generative models: Technologies, standards, and optimization," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 14, no. 2, pp. 149–171, 2024.

[37] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[38] E. Agustsson, D. Minnen, G. Toderici, and F. Mentzer, "Multi-realism image compression with a conditional generator," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 22 324–22 333.

[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[40] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[41] Z. Chen, L. Relic, R. Azevedo, Y. Zhang, M. Gross, D. Xu, L. Zhou, and C. Schroers, "Neural video compression with spatio-temporal cross-covariance transformers," in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 8543–8551.

[42] F. Mentzer, G. Toderici, D. Minnen, S.-J. Hwang, S. Caelles, M. Lucic, and E. Agustsson, "Vct: A video compression transformer," *arXiv preprint arXiv:2206.07307*, 2022.

[43] T. Ladune, P. Philippe, W. Hamidouche, L. Zhang, and O. Déforges, "Optical flow and mode selection for learning-based video coding," in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, IEEE, 2020, pp. 1–6.

[44] J. Li, B. Li, and Y. Lu, "Deep contextual video compression," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 114–18 125, 2021.

[45] X. Sheng, J. Li, B. Li, L. Li, D. Liu, and Y. Lu, "Temporal context mining for learned video compression," *IEEE Transactions on Multimedia*, vol. 25, pp. 7311–7322, 2022.

[46] J. Lin, D. Liu, J. Liang, H. Li, and F. Wu, "A deeply modulated scheme for variable-rate video compression," in *2021 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2021, pp. 3722–3726.

[47] Z. Hu and D. Xu, "Complexity-guided slimmable decoder for efficient deep video compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 14 358–14 367.

[48] J. Li, B. Li, and Y. Lu, "Hybrid spatial-temporal entropy modelling for neural video compression," in *Proceedings of the 30th ACM International Conference on Multimedia*, 2022, pp. 1503–1511.

[49] M. Lu, Z. Duan, F. Zhu, and Z. Ma, "Deep hierarchical video compression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, 2024, pp. 8859–8867.

[50] B. Liu, Y. Chen, R. C. Machineni, S. Liu, and H.-S. 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*, 2023, pp. 18 487–18 496.

[51] J. Li, B. Li, and Y. Lu, "Neural video compression with diverse contexts," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 22 616–22 626.

[52] L. Qi, J. Li, B. Li, H. Li, and Y. Lu, "Motion information propagation for neural video compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 6111–6120.

[53] J. Li, B. Li, and Y. Lu, "Neural video compression with feature modulation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 26 099–26 108.

[54] D. Alexandre, H.-M. Hang, and W.-H. 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*, 2023, pp. 10 249–10 258.

[55] J. Ballé, N. Johnston, and D. Minnen, "Integer networks for data compression with latent-variable models," in *International Conference on Learning Representations*, 2018.

[56] G. Gao, H. M. Kwan, F. Zhang, and D. Bull, "Pnvc: Towards practical inr-based video compression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, 2025, pp. 3068–3076.

[57] S. Ma, W. Gao, and Y. Lu, “Rate-distortion analysis for h. 264/avc video coding and its application to rate control,” *IEEE transactions on circuits and systems for video technology*, vol. 15, no. 12, pp. 1533–1544, 2005.

[58] D.-K. Kwon, M.-Y. Shen, and C.-C. J. Kuo, “Rate control for h. 264 video with enhanced rate and distortion models,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 5, pp. 517–529, 2007.

[59] Z. He, Y. K. Kim, and S. K. Mitra, “Low-delay rate control for dct video coding via/spl rho/-domain source modeling,” *IEEE transactions on Circuits and Systems for Video Technology*, vol. 11, no. 8, pp. 928–940, 2001.

[60] B. Li, J. Xu, D. Zhang, and H. Li, “Qp refinement according to lagrange multiplier for high efficiency video coding,” in *2013 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2013, pp. 477–480.

[61] A. Mercat, M. Viitanen, and J. Vanne, “Uvg dataset: 50/120fps 4k sequences for video codec analysis and development,” in *Proceedings of the 11th ACM multimedia systems conference*, 2020, pp. 297–302.

[62] H. Wang, W. Gan, S. Hu, J. Y. Lin, L. Jin, L. Song, P. Wang, I. Katsavounidis, A. Aaron, and C.-C. J. Kuo, “Mcl-jcv: A jnd-based h. 264/avc video quality assessment dataset,” in *2016 IEEE international conference on image processing (ICIP)*, IEEE, 2016, pp. 1509–1513.

[63] B. Gisle, “Calculation of average psnr differences between rd curves,” in *ITU-T SG16/Q6, 13^< th> VCEG Meeting, Austin, Texas, USA, April 2001*, 2001.