HCF: Hierarchical Cascade Framework for Distributed Multi-Stage Image Compression

Junhao Cai[1], Taegun An[1], Chengjun Jin[1], Sung Il Choi[1], Juhyun Park[1], Changhee Joo[1*]

[1]Korea University, Seoul, Republic of Korea

{junhochae,antaegun20,chengjunjin2001,sungchoi,juhyunpark,changhee}@korea.ac.kr

Abstract

Distributed multi-stage image compression—where visual content traverses multiple processing nodes under varying quality requirements—poses challenges. Progressive methods enable bitstream truncation but underutilize available compute resources; successive compression repeats costly pixel-domain operations and suffers cumulative quality loss and inefficiency; fixed-parameter models lack post-encoding flexibility. In this work, we developed the Hierarchical Cascade Framework (HCF) that achieves high rate-distortion performance and better computational efficiency through direct latent-space transformations across network nodes in distributed multi-stage image compression systems. Under HCF, we introduced policy-driven quantization control to optimize rate–distortion trade-offs, and established the edge quantization principle through differential entropy analysis. The configuration based on this principle demonstrates up to 0.6 dB PSNR gains over other configurations. When comprehensively evaluated on the Kodak, CLIC, and CLIC2020-mobile datasets, HCF outperforms successive-compression methods by up to 5.56 % BD-Rate in PSNR on CLIC, while saving up to 97.8 % FLOPs, 96.5 % GPU memory, and 90.0 % execution time. It also outperforms state-of-the-art progressive compression methods by up to 12.64 % BD-Rate on Kodak and enables retraining-free cross-quality adaptation with 7.13–10.87 % BD-Rate reductions on CLIC2020-mobile.

1 Introduction

The rapid growth of digital media consumption has significantly increased demands on visual information transmission systems (Ghosh and Singhal 2025; Cai, An, and Joo 2024), positioning image compression as an essential technology to address bandwidth and storage constraints while maintaining acceptable quality levels (Ibraheem, Dvorkovich, and Al-khafaji 2024). Modern transmission scenarios involve multi-stage processing where visual content traverses multiple processing nodes during transmission (Peroni and Gorinsky 2025). As deployment scenarios become increasingly complex, it is imperative for compression systems to provide multiple quality levels, enabling quality scaling during transmission (Kong and Sun 2024). This creates scenarios where compression operations are distributed across multiple processing stages with varying quality levels—a paradigm we refer to as *distributed multi-stage image compression*.

Within the distributed multi-stage compression framework, existing approaches have emerged but face fundamental limitations. Progressive compression methods (Lee et al. 2022; Jeon et al. 2023; Lee, Jeong, and Kim 2025; Presta et al. 2025) enable quality adaptation through bitstream truncation but constrain intermediate processing to passive operations with suboptimal rate-distortion trade-offs. Successive compression approaches (Kim et al. 2022) achieve adaptation through repeated operations but suffer from cumulative quality degradation and computational inefficiency due to redundant pixel-domain conversions. Meanwhile, fixed-parameter compression models (Minnen, Ballé, and Toderici 2018; Cheng et al. 2020; Jiang et al. 2025), while achieving superior rate-distortion performance in centralized settings, provide single operating points without post-encoding flexibility required for distributed multi-stage scenarios.

The emergence of distributed computational resources (Yang et al. 2025; Lin et al. 2024; Tang et al. 2024; Zheng et al. 2024; Kianpisheh and Taleb 2023) in modern networks creates new opportunities for intelligent quality transformation within distributed multi-stage compression, necessitating methods that can efficiently leverage these capabilities while avoiding the performance penalties of existing approaches. To address this need, we propose the Hierarchical Cascade Framework (HCF), which harnesses distributed computational resources for efficient multi-stage image compression. To the best of our knowledge, this is the first framework to effectively leverage distributed computational resources for multi-stage compression by enabling direct latent-space transformations across network nodes and implementing policy-driven quantization control. This approach simultaneously improves compression performance and reduces computational overhead while preventing cumulative degradation through optimal placement strategies.

The contributions of our work are as follows:

- We propose HCF, a novel framework facilitating distributed multi-stage image compression through direct latent-space transformations across network nodes, avoiding pixel-domain recompression cycles and introducing policy-driven quantization control.

*Corresponding Author

- We systematically analyze and identify limitations in existing distributed multi-stage compression approaches: progressive methods suffer from passive bitstream truncation without leveraging intermediate computational resources, while successive methods utilize computational resources but incur redundant pixel-domain processing cycles.
- We discover and establish the edge quantization optimality principle, providing theoretical insights into optimal quantization placement that consistently outperforms alternatives by up to 0.6 dB PSNR, validated through differential entropy analysis.
- We provide comprehensive evaluation across five compression architectures, achieving substantial improvements over existing state-of-the-art (SOTA) baselines: outperforming successive compression methods by up to 5.56% BD-Rate in PSNR and progressive compression methods by up to 12.64% BD-Rate, achieving 97.1–97.8% FLOPs reduction with 94.8–96.5% GPU memory savings and 77.9–90.0% execution time reduction, and enabling effective retraining-free cross-quality adaptation with 7.13–10.87% BD-Rate reductions.

2 Related Work

Image compression has evolved from traditional standards (Wallace 1992; Skodras, Christopoulos, and Ebrahimi 2001; Sullivan et al. 2012; Lian and Shilei 2012; Bellard 2014) to deep learning approaches that achieve superior rate-distortion performance. This paradigm shift was marked by key milestones: autoencoder architectures with neural transforms (Ballé, Laparra, and Simoncelli 2017; Theis et al. 2017), entropy modeling through hyperprior networks (Ballé et al. 2018), scale-mean Gaussians (Minnen, Ballé, and Toderici 2018), and mixture models (Cheng et al. 2020; Zhu et al. 2022; Fu et al. 2023), alongside context-aware processing that exploited local spatial dependencies (Minnen, Ballé, and Toderici 2018; He et al. 2022; Liu, Sun, and Katto 2023; Liu et al. 2024; Jiang et al. 2025), global correlations (Guo et al. 2021; Khoshkhahtinat et al. 2023), and channel-wise relationships (Feng et al. 2025; Zeng et al. 2025), as well as dictionary-based modeling (Wu et al. 2025; Lu et al. 2025) that leverages external priors. However, these learned methods remain designed as single-stage, centralized systems that generate static bitstreams, limiting their optimization potential in scenarios requiring post-encoding quality adaptation.

To address quality adaptation requirements, progressive compression approaches have evolved from early foundations with recurrent neural networks for variable compression rates (Toderici et al. 2016, 2017) and spatial adaptive bit allocation and diffusion mechanisms (Johnston et al. 2018). These advances enabled scalable coding architectures with layer-wise bit allocation (Su et al. 2020), and fine granular scalability through nested quantization (Lu et al. 2021), tri-plane encoding (Lee et al. 2022; Jeon et al. 2023), spatial autoregressive modeling with codeword alignment (Tian et al. 2023), multirate progressive entropy modeling that unifies split-coded-then-merge models (Li et al. 2024), progressive deep coding with layer-specific parameter learning (Lee, Jeong, and Kim 2025) and element-wise progressive transmission via variance-aware masking (Presta et al. 2025).

Meanwhile, successive image compression pipelines, where the same codec is repeatedly applied across stages via encoding–decoding cycles, can in principle produce multiple quality levels, but have been shown to suffer from cumulative quality degradation and instability (Kim et al. 2022). In parallel, the emergence of distributed computational resources in modern networks—including 6G Edge Networks (Lin et al. 2024), Mobile Edge Computing (Lee et al. 2024; Yang et al. 2025), and In-Network Machine Learning (Zheng et al. 2024)—highlights a broader shift towards exploiting intermediate nodes as active compute and decision points. These capabilities provide new opportunities for compression optimization through intermediate computational processing (Peroni and Gorinsky 2025). However, existing compression methods cannot effectively leverage such resources: progressive methods are constrained to suboptimal truncation operations with passive intermediate nodes, whereas successive approaches incur computational redundancy through repeated pixel-domain operations.

3 Methodology

3.1 Existing Frameworks and Limitations

Single-Stage Framework (SSF). Image compression systems traditionally operate under SSF, which follows a sequential pipeline comprising five fundamental operations. Let x denote the input image, $\tilde{y}$ the unquantized latent representation, $\hat{y}$ the quantized latent representation, $\mathcal{B}$ the bitstream, and $\hat{x}$ the reconstructed image. These operations proceed as

$$\tilde{y} = g_a(x), \quad \text{analysis transform,} \tag{1}$$
$$\hat{y} = Q(\tilde{y}), \quad \text{quantization,} \tag{2}$$
$$\mathcal{B} = E(\hat{y}), \quad \text{entropy encoding,} \tag{3}$$
$$\hat{y} = D(\mathcal{B}), \quad \text{entropy decoding,} \tag{4}$$
$$\hat{x} = g_s(\hat{y}), \quad \text{synthesis transform.} \tag{5}$$

Here, $g_a(\cdot)$, $Q(\cdot)$, $E(\cdot)$, $D(\cdot)$, and $g_s(\cdot)$ denote the analysis transform, quantization, entropy encoding, entropy decoding, and synthesis transform operators, respectively, all applied under a consistent quality configuration with rate–distortion parameter λ. While effective for centralized compression, SSF lacks post-encoding adaptability required for distributed multi-stage scenarios.

For distributed multi-stage compression, two primary approaches have emerged:

Progressive Compression Framework (PCF). PCF generates scalable bitstreams enabling quality adaptation through selective decoding and bitstream truncation. This "encode once, decode many" paradigm allows intermediate nodes to modify compression rates by discarding specific portions of the transmitted bitstream. However, this bitstream-centric approach restricts intermediate nodes to predetermined truncation patterns, without the flexibility to exploit node-specific computational resources for adaptive rate-distortion optimization in distributed multi-stage scenarios.

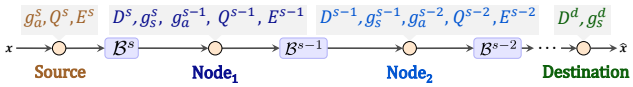

Figure 1: The Distributed Recompression Framework (DRF) extends SIC to distributed compression across multiple nodes. It repeats the decompression-compression cycle at intermediate nodes. Specifically, each Node$_k$ compresses the image at quality level $s-k$ followed by the full decompression operations of quality level $s-k+1$.

Distributed Recompression Framework (DRF). DRF extends Successive Image Compression (SIC) (Kim et al. 2022) to distributed multi-stage scenarios. Let s denote the source quality level, d the target quality level, with $s \geq d$, and let $k \in \{s, s-1, \ldots, d\}$ denote the stage index; a lower index implies a more compressed state. Under DRF, as shown in Figure 1, each intermediate node performs quality adaptation through pixel-level recompression: receiving compressed data at quality level $s-k+1$, applying decompression operations D^{s-k+1}, g_s^{s-k+1}, then compression operations $g_a^{s-k}, Q^{s-k}, E^{s-k}$ to achieve quality level $s-k$. Although it actively utilizes intermediate computational resources, it suffers from redundant pixel-domain conversions and computational inefficiency.

Fundamental Limitations. These frameworks face critical challenges in distributed multi-stage scenarios: PCF underutilizes available computational resources and DRF incurs computational inefficiency through redundant processing cycles. Both approaches treat quality adaptation as separate from the compression process itself, limiting optimization potential for distributed multi-stage compression.

3.2 Hierarchical Cascade Framework

To address these fundamental limitations, we propose the Hierarchical Cascade Framework (HCF) that introduces a paradigm shift from adaptation-after-compression to adaptation-during-compression. We assume that the quality levels are ordered with $\lambda_s > \lambda_{s-1} > \cdots > \lambda_d$, where λ_k denotes the rate-distortion parameter for quality level k, and higher λ values correspond to higher compression quality. Instead of treating distributed nodes as passive relay points or applying pixel-domain recompression cycles, HCF establishes direct latent-space transformation pathways that preserve and enhance compression efficiency throughout the distributed pipeline.

The framework introduces two key innovations: direct latent-space transformation and selective quantization with policy-driven control. Upon receiving compressed data, an intermediate node may or may not perform compression. For ease of exposition, we focus only on the nodes that perform compression, omitting the others. This design allows a node to compress the received image by multiple levels, depending on available computational resources. Note that if a node compresses the received data by more than two quality levels (e.g., from level 6 directly to level 3), intermediate quantization and entropy encoding operations can be skipped since only the final quantization is transmitted. The analysis transform (g_a) and synthesis transform (g_s) are applied only at the beginning and end stages, while intermediate nodes apply direct transformations to the latent space. These changes not only simplify the whole operations, but also allow us to selectively conduct the operations at intermediate nodes. Under this framework, we are able to dynamically optimize the overall process of multi-stage compression over the path according to resource availability at the nodes.

We start by defining the process type. From the perspective of the compression operations, we categorize the level-k process into two different types: inter-node process that includes quantization and entropy encoding, and intra-node process that does not.

The *inter-node process* is used when data transmission to the next hop is involved during the level-k process. It consists of the quantization Q^k, entropy encoding E^k, and entropy decoding D^k operations, followed by transform module $\phi^{\text{inter}}_{k\to k-1}$. The transmission occurs between the entropy encoding and decoding operations. Given unquantized input $\tilde{y}^k$, we present the level-k inter-node process as

$$\mathcal{T}_k^{\text{inter}}(\tilde{y}^k) := \left(\phi^{\text{inter}}_{k\to k-1} \circ D^k \circ E^k \circ Q^k\right)(\tilde{y}^k), \quad (6)$$

where $\circ$ denotes function composition. The transform module $\phi^{\text{inter}}_{k\to k-1}$ is specifically designed for 'quantized' input.

The *intra-node process*, in contrast, simply preserves the unquantized input $\tilde{y}^k$ and directly applies a transform module designed for 'unquantized' input. We present the level-k intra-node process as

$$\mathcal{T}_k^{\text{intra}}(\tilde{y}^k) := \phi^{\text{intra}}_{k\to k-1}(\tilde{y}^k). \quad (7)$$

Note that both the inter- and intra-node processes end with a transform module that shares an identical architecture (see Figure 2(b)) but requires separate training for different input distributions at each level. The novel modules contain residual blocks, attention mechanisms, convolution operations with GDN (Ballé, Laparra, and Simoncelli 2016) and LeakyReLU activations, and channel dimension adjustments to handle both quantized and unquantized latent representations. Our primary contribution lies in the development of a systematic framework enabling policy-driven quantization control across multiple compression stages, with the unified architecture facilitating seamless integration across the compression pipeline. Ablation studies (Section 4.5) demonstrate the effectiveness of this design and confirm superior performance in distributed multi-stage scenarios.

The separation of the inter- and intra-node processes allows us to selectively quantize the data at each level k, and optimize the compression operations over the network. We define the *policy vector* $\boldsymbol{\pi} = [\pi_s, \pi_{s-1}, \ldots, \pi_d]$ where $\pi_k \in \{0, 1\}$ indicates the type of level k process: $\pi_k = 1$ for the inter-node process and $\pi_k = 0$ for the intra-node process. We fix $\pi_d = 1$ at the last level since the level-d data should be transmitted to the destination and will be decompressed.

Using π_k, we can present the level-k process in a unified manner as

$$\mathcal{T}_k^{\pi_k}(\tilde{y}^k) = (1-\pi_k)\mathcal{T}_k^{\text{intra}}(\tilde{y}^k) + \pi_k \mathcal{T}_k^{\text{inter}}(\tilde{y}^k). \quad (8)$$

We define the transformation under policy $\boldsymbol{\pi}$ from level s to any intermediate level k as the composition $\mathcal{F}^{\boldsymbol{\pi}}_{s\to k} =$

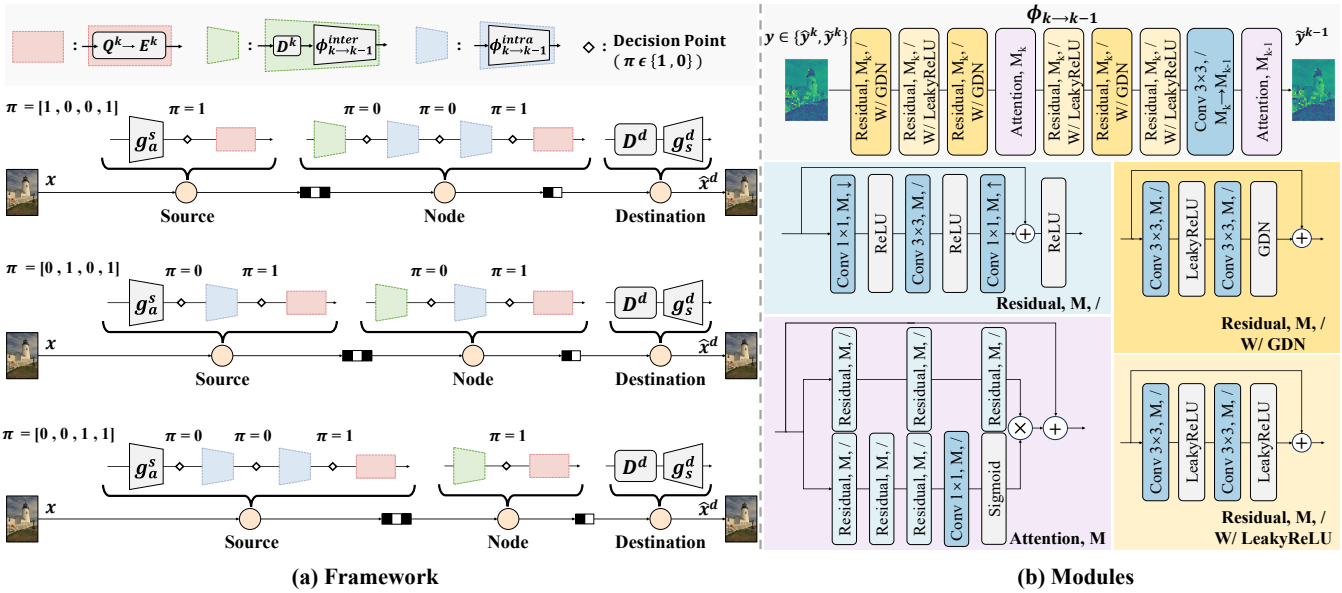

Figure 2: Hierarchical Cascade Framework (HCF). (a) Three 4-level compression examples in a 2-stage compression system, each corresponding to a different policy vector $\boldsymbol{\pi} \in \{[1,0,0,1],[0,1,0,1],[0,0,1,1]\}$ that determines the quantization placement in the transformation chain. At each level k, the process type follows the definition in Eq. (8). (b) Architecture of transform modules $\phi_{k\to k-1}$ designed for cascade processing. M denotes the number of channels in the latent representation. In convolutions, the change of the channel dimension is denoted by $\downarrow$ ($\times\frac{1}{2}$), $\uparrow$ ($\times 2$), and / (unchanged), respectively. The notation $M_k \to M_{k-1}$ indicates that the convolution maps the latent features from level k to level $k-1$. All convolutions have stride 1.

$\mathcal{T}_{k+1}^{\pi_{k+1}} \circ \cdots \circ \mathcal{T}_s^{\pi_s}$, and the complete transformation from source to destination as

$$\mathcal{F}_{s\to d}^{\boldsymbol{\pi}} = \mathcal{T}_{d+1}^{\pi_{d+1}} \circ \mathcal{T}_{d+2}^{\pi_{d+2}} \circ \cdots \circ \mathcal{T}_{s-1}^{\pi_{s-1}} \circ \mathcal{T}_s^{\pi_s}. \quad (9)$$

Including the initial process at the source, the last level-d process (without a transform module), and the decompression at the destination, the complete end-to-end cascade process $\mathcal{C}(s,d,\boldsymbol{\pi}) : x \mapsto \hat{x}^d$ mapping input image x to reconstruction $\hat{x}^d$ can be represented as

$$\mathcal{C}(s,d,\boldsymbol{\pi})(x) = \big(g_s^d \circ D^d \circ E^d \circ Q^d \circ \mathcal{F}_{s\to d}^{\boldsymbol{\pi}} \circ g_a^s\big)(x). \quad (10)$$

Figure 2 provides a comprehensive visual representation of our HCF framework in a simple network scenario with one intermediate node between the source and destination. Both nodes participate in a 4-level compression process, forming a 2-stage compression system. Accordingly, the policy vector $\boldsymbol{\pi}$ has length 4 with $\pi_d = 1$. In this 2-stage setting, two transmissions occur along the path, and each transmission must terminate with a quantization operation. HCF enables strategic placement of these quantization points within the overall compression pipeline. For a 4-level, 2-stage configuration, there are three possible quantization placements, $\boldsymbol{\pi} \in \{[1,0,0,1],[0,1,0,1],[0,0,1,1]\}$, since the final level-$d$ process necessarily includes transmission to the destination. As the leftmost 1 in $\boldsymbol{\pi}$ shifts rightward, a larger portion of the compression workload is handled by the source, while the intermediate node processes fewer levels.

3.3 Training Strategy

Our training strategy consists of two sequential phases: transform module training and fine-tuning. We utilize pre-trained single-stage compression model parameters.

Training of transform modules. We sequentially train transform modules $\phi_{k\to k-1}^{\text{inter}}$ or $\phi_{k\to k-1}^{\text{intra}}$ for each level pair $(k, k-1)$ from $k=s$ down to $k=d+1$. For intra-node modules (unquantized input), we minimize

$$\mathcal{L}_{\text{intra}}^{k\to k-1} = \big\|\phi_{k\to k-1}^{\text{intra}}(\tilde{y}^k) - g_a^{k-1}(x)\big\|_2^2, \quad (11)$$

and for inter-node modules (quantized input), we minimize

$$\mathcal{L}_{\text{inter}}^{k\to k-1} = \big\|\phi_{k\to k-1}^{\text{inter}}(\hat{y}^k) - g_a^{k-1}(x)\big\|_2^2, \quad (12)$$

where $\tilde{y}^k = (\mathcal{F}_{s\to k}^{\boldsymbol{\pi}} \circ g_a^s)(x)$ and $\hat{y}^k = D^k\big(E^k\big(Q^k(\tilde{y}^k)\big)\big)$. During training, all previously optimized networks are frozen, and only the current transform module is updated. Our unified architecture handles both quantized and unquantized inputs via weight switching rather than separate models, minimizing deployment overhead. Inspired by knowledge distillation (Hinton, Vinyals, and Dean 2015), our modules enable adaptive compression across distributed quality levels in multi-stage compression systems.

Fine-tuning. We perform end-to-end optimization for each level $k \in \{s, s-1, \ldots, d\}$ using the rate–distortion objective:

$$\mathcal{L}_{\text{RD}}^k = \lambda_k \cdot \mathcal{D}(x,\hat{x}^k) + \mathcal{R}(\hat{y}^k), \quad (13)$$

where $\tilde{y}^k = \big(\mathcal{F}_{s\to k}^{\boldsymbol{\pi}} \circ g_a^s\big)(x)$, $\hat{y}^k = D^k\big(E^k\big(Q^k(\tilde{y}^k)\big)\big)$, and $\hat{x}^k = g_s^k(\hat{y}^k)$. Here, $\mathcal{D}(\cdot,\cdot)$ and $\mathcal{R}(\cdot)$ denote the distortion and rate estimation functions, respectively. We freeze networks from higher quality levels and all transform modules, optimizing only the synthesis transform g_s^k and associated entropy models at level k.

Quant. Freq. (n_q)	Target Qual. (d)	Policy Vector (π)	Bitrate (bpp)	PSNR ↑	MS-SSIM ↑	η^{PSNR} ↓	$\eta^{\text{MS-SSIM}}$ ↓
2	2	**[1,0,0,0,1]**	**0.1674**	**30.264**	**13.129**	**11.054**	**13.573**
		[0,0,0,1,1]	0.1674	29.768	12.761	17.115	18.070
4	1	**[1,1,1,0,0,1]**	**0.0981**	**28.524**	**11.249**	**19.057**	**22.361**
		[0,0,1,1,1,1]	0.0951	27.929	10.819	27.419	28.010

Table 1: Representative quantization policy comparison using MLIC++ with HCF on Kodak dataset. Bold entries show consistently superior performance of $\boldsymbol{\pi}^{\text{edge}}$ strategies (↑ higher is better, ↓ lower is better). Complete analysis across all quantization frequencies, configurations, and additional datasets in Supplementary Material.

4 Experiments

4.1 Experimental Setup

We validated HCF across five diverse compression architectures—MLIC++ (Jiang et al. 2025), cheng2020_attn and cheng2020_anchor (Cheng et al. 2020), and mbt2018 and mbt2018_mean (Minnen, Ballé, and Toderici 2018)—spanning from hyperprior models to context-attention mechanisms, ensuring generalization across architectural paradigms. We train the models following our two-phase strategy (Section 3.3) using DUTS dataset (Wang et al. 2017) with standard augmentation. The models are initialized from pre-trained CompressAI (Bégaint et al. 2020) weights and official implementations with quality levels $s = 6$ (MLIC++, cheng2020_anchor, cheng2020_attn) and $s = 8$ (mbt2018, mbt2018_mean). We use Kodak (Kodak 1993), CLIC2020-mobile (Toderici et al. 2020), and CLIC (Toderici et al. 2020) datasets for the evaluation.

We systematically compared HCF against four baseline categories: (1) PCF including latest SOTA approaches (Presta (Presta et al. 2025), DeepHQ (Lee, Jeong, and Kim 2025), Li (Li et al. 2024), Jeon (Jeon et al. 2023), Tian (Tian et al. 2023), Lee (Lee et al. 2022)) and foundational works (Toderici et al. 2017; Johnston et al. 2018; Diao, Ding, and Tarokh 2020); (2) DRF extending successive compression (Kim et al. 2022); (3) SSF as performance bounds; (4) Traditional standards (JPEG (Wallace 1992), JPEG2000 (Skodras, Christopoulos, and Ebrahimi 2001), BPG (Bellard 2014), HEVC (Sullivan et al. 2012), WebP (Lian and Shilei 2012)).

For a fair comparison, DRF and SSF use identical architectures to HCF. We report rate–distortion curves and Bjøntegaard Delta metrics (Bjøntegaard 2001): BD-Rate$_\text{P}$/BD-Rate$_\text{M}$ (bitrate reduction for PSNR/MS-SSIM (Wang, Simoncelli, and Bovik 2003), lower is better) and BD-PSNR/BD-MS-SSIM (quality improvement, higher is better), along with computational efficiency metrics including FLOPs, parameters, GPU memory, and execution time reduction relative to the baselines. All experiments used NVIDIA RTX A6000 GPUs. The experimental details and hyperparameters are described in Supplementary Material.

4.2 Policy Analysis

HCF's flexibility in quantization placement across multiple compression stages naturally raises questions about optimal policy selection. To address this systematically, we establish a mathematical framework for policy representation. For an n_q-quantization compression system (where n_q denotes the number of quantization operations) with quality levels from s to d, we define the policy space as all feasible quantization placement patterns with exactly n_q quantization operations, which is formally given by

$$\Pi(s, d; n_q) := \left\{ \boldsymbol{\pi} \in \{0,1\}^{s-d+1} \mid \sum_{k=d}^{s} \boldsymbol{\pi}_k = n_q,\ \pi_d = 1 \right\}.$$

Given the significant variation in rate-distortion performance across different policies within the same n_q constraint, we evaluated policy effectiveness using established quality metrics (e.g., PSNR, MS-SSIM). To gain deeper insights into the fundamental principles governing optimal quantization placement, we developed a Rate-Quality Sensitivity Index (RQSI) for analyzing quantization efficiency.

Rate-Quality Sensitivity Index (RQSI). We quantified policy efficiency using reference vectors: $\boldsymbol{\pi}_* = [0, \ldots, 0, 1] \in \Pi(s, d; 1)$ (minimal quantization, only at the final level) and $\boldsymbol{\pi}^* = [1, \ldots, 1, 1] \in \Pi(s, d; s-d+1)$ (maximal quantization, at every level). Let $\mathcal{M}$ denote the quality metric (i.e., PSNR or MS-SSIM), and let $\hat{x}^{d+1}_{\boldsymbol{\pi}}$, $\hat{x}^{d}_{\boldsymbol{\pi}}$ and $\hat{y}^{d+1}_{\boldsymbol{\pi}}$, $\hat{y}^{d}_{\boldsymbol{\pi}}$ denote reconstructed images and quantized latent representations at quality levels $d+1$ and d under policy $\boldsymbol{\pi}$. We define the RQSI of policy $\boldsymbol{\pi}$ as

$$\eta^{\mathcal{M}}(\boldsymbol{\pi}) = \tfrac{1}{2}\left(RQS(\boldsymbol{\pi}, \boldsymbol{\pi}_*) + RQS(\boldsymbol{\pi}, \boldsymbol{\pi}^*)\right), \tag{14}$$

where

$$RQS(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2) = \frac{\left|\mathcal{M}(x, \hat{x}^{d+1}_{\boldsymbol{\pi}_2}) - \mathcal{M}(x, \hat{x}^{d}_{\boldsymbol{\pi}_1})\right|}{\max\left(\left|\mathcal{R}(\hat{y}^{d+1}_{\boldsymbol{\pi}_2}) - \mathcal{R}(\hat{y}^{d}_{\boldsymbol{\pi}_1})\right|, \varepsilon\right)}, \tag{15}$$

and $\varepsilon > 0$ is a small constant ensuring numerical stability (e.g., $\varepsilon = 10^{-6}$). The RQSI metric $\eta^{\mathcal{M}}$ provides a measure to compare quantization placement strategies, with lower values indicating superior rate-quality efficiency. By benchmarking against $\boldsymbol{\pi}_*$ and $\boldsymbol{\pi}^*$, RQSI isolates the impact of quantization placement within the cascade, independent of the compression level. This identifies policies that preserve critical information for downstream transformations.

We conducted comprehensive experiments using MLIC++ on the Kodak dataset across different quantization frequencies. Table 1 presents representative comparisons between two different policies, with bold entries indicating superior performance. For $n_q = 2$ and target quality $d = 2$, the policy $[1, 0, 0, 0, 1]$ achieved 30.264 dB PSNR compared to 29.768 dB for the alternative policy $[0, 0, 0, 1, 1]$ at identical bitrates (0.50 dB improvement). For the more

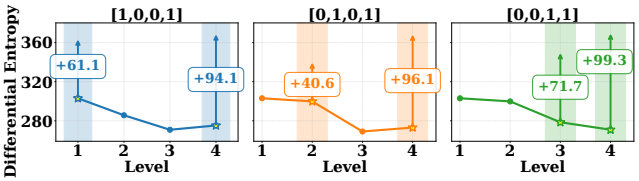

Figure 3: Differential entropy evolution across compression levels for HCF's three quantization policies using MLIC++ on Kodak in a 2-stage, 4-level system. Stars indicate quantization points; ↑ show entropy increases after quantization.

complex scenario with $n_q = 4$ and $d = 1$, the policy $[1, 1, 1, 0, 0, 1]$ achieves 28.524 dB PSNR versus 27.929 dB for $[0, 0, 1, 1, 1, 1]$ (0.60 dB gain) while using only marginally higher bitrate (0.0981 vs 0.0951 bpp). Comprehensive results across all quantization frequencies (n_q) and target quality levels (d) are in Supplementary Material.

In these examples, we can observe that early-quantization strategies–those that '1's appear at early stage, i.e., $[1, 0, 0, 0, 1]$ and $[1, 1, 1, 0, 0, 1]$–outperform their counterparts. This, indeed, has been observed throughout our experiments. The RQSI metric further clarifies the observation by showing that the edge policy achieves a lower RQSI value: $\eta^{\text{PSNR}}([1, 0, 0, 0, 1]) = 11.054 < \eta^{\text{PSNR}}([0, 0, 0, 1, 1]) = 17.115$ and $\eta^{\text{PSNR}}([1, 1, 1, 0, 0, 1]) = 19.057 < \eta^{\text{PSNR}}([0, 0, 1, 1, 1, 1]) = 27.419$.

We formally define such early-quantization policy as *edge policy* π^{edge} as

$$\boldsymbol{\pi}^{\text{edge}} = [1^{(n_q-1)}, 0^{(s-d+1-n_q)}, 1], \tag{16}$$

where 1^k and 0^k denote k consecutive ones and zeros (here $k = n_q - 1$ and $k = s - d + 1 - n_q$ respectively). This strategically places quantization at cascade edges: $(n_q - 1)$ operations at front stages plus one mandatory final operation. Evaluation across all quantization frequencies and configurations (detailed in Supplementary Material) consistently validated this edge quantization principle.

To understand the underlying mechanism driving this performance difference, we analyzed differential entropy evolution across compression levels using the Kozachenko-Leonenko estimator (Kozachenko and Leonenko 1987). Figure 3 shows differential entropy evolution across four compression levels for three quantization policies. While all policies converge to similar entropy values before the final quantization operation, they exhibit markedly different entropy increases after quantization: 94.1, 96.1, and 99.3 bits for policies [1,0,0,1], [0,1,0,1], and [0,0,1,1], respectively. Policy $\boldsymbol{\pi}^{\text{edge}} = [1, 0, 0, 1]$ achieved the smallest increment, demonstrating that early quantization injection allows subsequent transform modules to more effectively decorrelate and suppress redundant information, thereby minimizing the irreducible uncertainty that must be encoded. This theoretical advantage translated into visually perceptible improvements, as demonstrated in Figure 4, where $\boldsymbol{\pi}^{\text{edge}}$ preserves significantly more structural and textural details compared to alternative policies at equivalent bitrates. Comprehensive entropy analysis across different quality configurations (detailed in Supplementary Material) confirms that this pattern consistently holds across diverse scenarios.

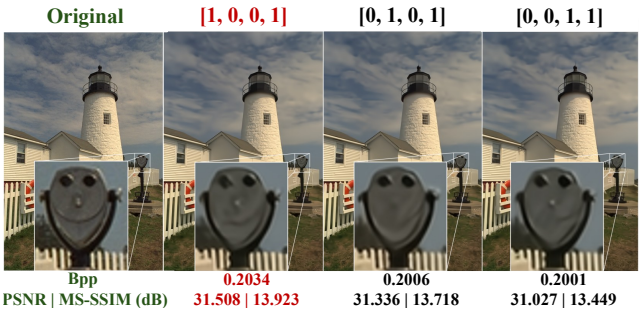

Figure 4: Visual quality comparison of HCF policies on Kodak dataset using MLIC++. $\boldsymbol{\pi}^{\text{edge}} = [1, 0, 0, 1]$ preserves more details compared to alternatives at equivalent bitrates.

Model	Kodak		CLIC	
	BD-Rate$_P$ (%) ↓	BD-PSNR (dB) ↑	BD-Rate$_P$ (%) ↓	BD-PSNR (dB) ↑
Presta	+12.64	-0.46	+9.48	-0.28
Jeon	+13.40	-0.51	+8.62	-0.27
Lee	+43.84	-1.76	+26.39	-0.80
DRF	+4.87	-0.22	+5.56	-0.23

Table 2: BD-metric evaluation relative to HCF with $\boldsymbol{\pi}^{\text{edge}}$ policy using MLIC++. DRF is the MLIC++ DRF variant. Others are PCF variants. ↑ higher is better, ↓ lower is better.

4.3 Performance Comparison

Based on the superior $\boldsymbol{\pi}^{\text{edge}}$ policy identified through systematic analysis across multiple quantization frequencies, we evaluated HCF's overall performance against established baselines. We focus on the widely-applicable $n_q = 2$ configuration, which represents the most common two-stage compression scenario in practical distributed systems (e.g., source → base station → destination), while maintaining computational tractability. This configuration consistently demonstrates the edge quantization principle across all tested architectures, with similar performance patterns observed for higher quantization frequencies ($n_q > 2$) as detailed in Supplementary Material.

Figure 5 presents comprehensive rate-distortion comparisons across five compression architectures: SSF (centralized baseline), DRF (distributed recompression baseline), PCF variants (bitstream truncation), and HCF using $\boldsymbol{\pi}^{\text{edge}}$. HCF demonstrates superior performance over distributed baselines while approaching centralized SSF performance, with consistent cross-architecture improvements evidencing our approach's generalizability. Table 2 quantifies these improvements using MLIC++ architecture under fair comparison conditions—relative to HCF with MLIC++ using $\boldsymbol{\pi}^{\text{edge}}$ policy, PCF methods exhibited BD-Rate$_P$ increases of 12.64% (Presta), 13.40% (Jeon), and 43.84% (Lee) on Kodak with similar CLIC trends, while DRF shows 4.87%

Model	*FLOPs Reduction / Parameters Reduction / GPU Memory Reduction / Execution Time Reduction (%)* ↑				
	Node$_1$	**Node$_2$**	**Node$_3$**	**Node$_4$**	**Node$_5$**
cheng2020_attn	97.8/68.7/95.5/90.0	97.8/68.7/95.5/90.0	97.4/65.8/95.0/87.6	97.8/68.7/96.5/83.5	97.8/68.7/96.5/83.5
cheng2020_anchor	97.5/62.7/95.4/87.1	97.5/62.7/95.4/87.1	97.1/60.2/94.8/83.9	97.5/62.7/96.5/77.9	97.5/62.7/96.5/77.9

Table 3: Computational efficiency comparison: HCF vs. DRF across five processing nodes. Values show HCF's percentage reduction in FLOPs/Parameters/GPU Memory/Execution Time relative to DRF for transform operations. ↑ higher is better.

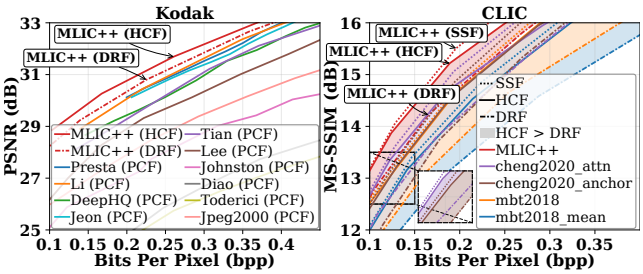

Figure 5: Rate-distortion comparison using PSNR on Kodak dataset and MS-SSIM on CLIC dataset. HCF with π^{edge} policy outperforms PCF and DRF baselines while approaching SSF performance. Shaded areas indicate HCF's performance advantage over DRF. Upper-left is better.

(Kodak) and 5.56% (CLIC) increases, demonstrating direct latent-space transformations' effectiveness over bitstream truncation and pixel-domain recompression. Beyond rate-distortion gains, HCF achieved substantial computational efficiency through eliminating redundant decode-encode operations: Table 3 shows overall reductions of 97.1–97.8% FLOPs, 94.8–96.5% GPU memory, and 77.9–90.0% execution time across evaluated models. Additional results across datasets and architectures are in Supplementary Material.

4.4 Cross-Quality Adaptation and Error Control

HCF enables retraining-free adaptation across quality levels through learned latent-space transformations. These transformations provide error control across compression trajectories. Our framework supports multiple compression paths using a single model, eliminating the need for separate training for each quality configuration. Table 4 demonstrates this using cheng2020_attn on CLIC2020-mobile, where shorter compression paths (e.g., $5 \rightarrow 1$) achieve better BD-Rate$_\text{P}$ reductions by 7.13–10.87% over longer paths (e.g., $6 \rightarrow 1$) to the same target quality. This validated HCF's effective error accumulation control—longer spans introduce more cumulative distortion, while our learned transformations mitigate degradation without retraining, with additional results across architectures provided in Supplementary Material.

4.5 Ablation Studies

To validate our design hypothesis that both transform modules are essential and complementary, we compared three HCF configurations: complete HCF ($\phi^{\text{inter}} + \phi^{\text{intra}}$), ϕ^{inter}-only, and ϕ^{intra}-only. All use π^{edge} policy. Figure 6 shows that removing either module leads to performance degradation: ϕ^{inter}-only suffered at lower bitrates due to inadequate information preservation, while ϕ^{intra}-only fails at higher bitrates due to poor quantization artifact handling. These results confirmed that both modules address distinct challenges and their combination is essential for superior performance, with additional ablation studies in Supplementary.

Model	Compression Path Comparison	BD-Rate$_\text{P}$ (%) ↓	BD-Rate$_\text{M}$ (%) ↓
cheng2020_attn	$5 \rightarrow 1$ vs. $6 \rightarrow 1$	-7.13	-7.29
	$4 \rightarrow 1$ vs. $5 \rightarrow 1$	-7.36	-7.02
	$3 \rightarrow 1$ vs. $4 \rightarrow 1$	-10.87	-7.80

Table 4: HCF cross-quality adaptation without retraining on CLIC2020-mobile dataset. BD-Rate reductions validate controlled error accumulation. ↓ lower is better.

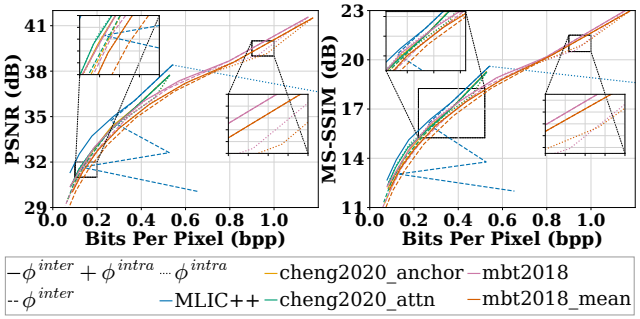

Figure 6: Ablation study comparing complete HCF ($\phi^{\text{inter}} + \phi^{\text{intra}}$) with ϕ^{inter}-only and ϕ^{intra}-only variants on CLIC2020-mobile dataset under π^{edge} policy. Both modules are essential for superior performance. Upper-left is better.

5 Conclusion

We propose Hierarchical Cascade Framework (HCF), a novel framework enabling direct latent-space transformations for distributed multi-stage image compression. The framework introduces policy-driven quantization control with optimal strategies (π^{edge}) that outperform SOTA methods by up to 12.64% BD-Rate in PSNR and improve computational efficiency (e.g., FLOPs reduction up to 97.8%). HCF delivers superior rate-distortion performance while enabling retraining-free cross-quality adaptation. This work contributes to establishing a new paradigm for learned compression across transmission stages in intelligent multimedia systems. Future extensions will target video compression scenarios and intelligent agents for adaptive policy control.

Acknowledgments

This work is supported in part by NRF grant funded by the Korea government (MSIT) (RS-2022-NR070834, 33%), by IITP grant funded by the Korea government (MSIT) (RS-2024-00405128, 33%), and by the ICT Creative Consilience program through IITP grant funded by the Korea government (MSIT) (IITP-2025-RS-2020-II201819, 33%).

References

Ballé, J.; Laparra, V.; and Simoncelli, E. P. 2016. Density modeling of images using a generalized normalization transformation. In *4th International Conference on Learning Representations, ICLR 2016*.

Ballé, J.; Laparra, V.; and Simoncelli, E. P. 2017. End-to-end Optimized Image Compression. In *International Conference on Learning Representations*.

Ballé, J.; Minnen, D.; Singh, S.; Hwang, S. J.; and Johnston, N. 2018. Variational image compression with a scale hyperprior. In *International Conference on Learning Representations*.

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

Bellard, F. 2014. BPG Image Format. https://bellard.org/bpg/.

Bjøntegaard, G. 2001. Calculation of Average PSNR Differences between RD-curves. *ITU-T VCEG-M33*.

Cai, J.; An, T.; and Joo, C. 2024. Toward Scalable and Efficient Visual Data Transmission in 6G Networks. In *2024 15th International Conference on Information and Communication Technology Convergence (ICTC)*, 545–547. IEEE.

Cheng, Z.; Sun, H.; Takeuchi, M.; and Katto, J. 2020. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 7939–7948.

Diao, E.; Ding, J.; and Tarokh, V. 2020. DRASIC: Distributed Recurrent Autoencoder for Scalable Image Compression. In *2020 Data Compression Conference (DCC)*, 3–12.

Feng, D.; Cheng, Z.; Wang, S.; Wu, R.; Hu, H.; Lu, G.; and Song, L. 2025. Linear Attention Modeling for Learned Image Compression. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 7623–7632.

Fu, H.; Liang, F.; Lin, J.; Li, B.; Akbari, M.; Liang, J.; Zhang, G.; Liu, D.; Tu, C.; and Han, J. 2023. Learned image compression with gaussian-laplacian-logistic mixture model and concatenated residual modules. *IEEE Transactions on Image Processing*, 32: 2063–2076.

Ghosh, M.; and Singhal, C. 2025. A review on machine learning based user-centric multimedia streaming techniques. *Computer Communications*, 231: 108011.

Guo, Z.; Zhang, Z.; Feng, R.; and Chen, Z. 2021. Causal contextual prediction for learned image compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(4): 2329–2341.

He, D.; Yang, Z.; Peng, W.; Ma, R.; Qin, H.; and Wang, Y. 2022. Elic: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5718–5727.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Ibraheem, M. K. I.; Dvorkovich, A. V.; and Al-khafaji, I. M. A. 2024. A Comprehensive Literature Review on Image and Video Compression: Trends, Algorithms, and Techniques. *Ingénierie des Systèmes d'Information*, 29(3).

Jeon, S.; Choi, K. P.; Park, Y.; and Kim, C.-S. 2023. Context-Based Trit-Plane Coding for Progressive Image Compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Jiang, W.; Yang, J.; Zhai, Y.; Gao, F.; and Wang, R. 2025. MLIC++: Linear Complexity Multi-Reference Entropy Modeling for Learned Image Compression. *ACM Trans. Multimedia Comput. Commun. Appl.*

Johnston, N.; Vincent, D.; Minnen, D.; Covell, M.; Singh, S.; Chinen, T.; Jin Hwang, S.; Shor, J.; and Toderici, G. 2018. Improved Lossy Image Compression with Priming and Spatially Adaptive Bit Rates for Recurrent Networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4385–4393.

Khoshkhahtinat, A.; Zafari, A.; Mehta, P. M.; Akyash, M.; Kashiani, H.; and Nasrabadi, N. M. 2023. Multi-context dual hyper-prior neural image compression. In *2023 International Conference on Machine Learning and Applications (ICMLA)*, 618–625. IEEE.

Kianpisheh, S.; and Taleb, T. 2023. A Survey on In-Network Computing: Programmable Data Plane and Technology Specific Applications. *IEEE Communications Surveys & Tutorials*, 25(1): 701–761.

Kim, J.-H.; Jang, S.; Choi, J.-H.; and Lee, J.-S. 2022. Successive learned image compression: Comprehensive analysis of instability. *Neurocomputing*, 506: 12–24.

Kodak, E. 1993. Kodak lossless true color image suite (PhotoCD PCD0992).

Kong, W.; and Sun, M. 2024. Progressive image compression for Gaussian mixture model quartile intervals. *Applied Intelligence*, 54(15): 7493–7508.

Kozachenko, L. F.; and Leonenko, N. N. 1987. Sample Estimate of the Entropy of a Random Vector. *Problems of Information Transmission*, 23(2): 95–101. Original Russian version: *Probl. Peredachi Inf.*, 23(2):9–16, 1987.

Lee, H.; Il Choi, S.; Hyun Lee, S.; Debbah, M.; and Lee, I. 2024. Distributed Task Offloading in Mobile-Edge Computing With Virtual Machines. *IEEE Internet of Things Journal*, 11(13): 24083–24097.

Lee, J.; Jeong, S. Y.; and Kim, M. 2025. DeepHQ: Learned Hierarchical Quantizer for Progressive Deep Image Coding. *ACM Trans. Multimedia Comput. Commun. Appl.*

Lee, J.-H.; Jeon, S.; Choi, K. P.; Park, Y.; and Kim, C.-S. 2022. DPICT: Deep Progressive Image Compression Using Trit-Planes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Li, C.; Yin, S.; Jia, C.; Meng, F.; Tian, Y.; and Liang, Y. 2024. Multirate Progressive Entropy Model for Learned Image Compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 34(8): 7725–7741.

Lian, L.; and Shilei, W. 2012. Webp: A new image compression format based on vp8 encoding. *Microcontrollers & Embedded Systems*, 3: 2.

Lin, Z.; Qu, G.; Chen, X.; and Huang, K. 2024. Split learning in 6G edge networks. *IEEE Wireless Communications*.

Liu, J.; Sun, H.; and Katto, J. 2023. Learned image compression with mixed transformer-cnn architectures. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 14388–14397.

Liu, Y.; Yang, W.; Bai, H.; Wei, Y.; and Zhao, Y. 2024. Region-adaptive transform with segmentation prior for image compression. In *European Conference on Computer Vision*, 181–197. Springer.

Lu, J.; Zhang, L.; Zhou, X.; Li, M.; Li, W.; and Gu, S. 2025. Learned Image Compression with Dictionary-based Entropy Model. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 12850–12859.

Lu, Y.; Zhu, Y.; Yang, Y.; Said, A.; and Cohen, T. S. 2021. Progressive neural image compression with nested quantization and latent ordering. In *2021 IEEE International conference on image processing (ICIP)*, 539–543. IEEE.

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

Peroni, L.; and Gorinsky, S. 2025. An End-to-End Pipeline Perspective on Video Streaming in Best-Effort Networks: A Survey and Tutorial. *ACM Comput. Surv.*, 57(12).

Presta, A.; Tartaglione, E.; Fiandrotti, A.; Grangetto, M.; and Cosman, P. 2025. Efficient Progressive Image Compression with Variance-Aware Masking. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*.

Skodras, A.; Christopoulos, C.; and Ebrahimi, T. 2001. The JPEG 2000 still image compression standard. *IEEE Signal processing magazine*, 18(5): 36–58.

Su, R.; Cheng, Z.; Sun, H.; and Katto, J. 2020. Scalable Learned Image Compression With A Recurrent Neural Networks-Based Hyperprior. In *2020 IEEE International Conference on Image Processing (ICIP)*, 3369–3373.

Sullivan, G. J.; Ohm, J.-R.; Han, W.-J.; and Wiegand, T. 2012. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12): 1649–1668.

Tang, S.; Yu, Y.; Wang, H.; Wang, G.; Chen, W.; Xu, Z.; Guo, S.; and Gao, W. 2024. A Survey on Scheduling Techniques in Computing and Network Convergence. *IEEE Communications Surveys & Tutorials*, 26(1): 160–195.

Theis, L.; Shi, W.; Cunningham, A.; and Huszár, F. 2017. Lossy Image Compression with Compressive Autoencoders. In *International Conference on Learning Representations*.

Tian, W.; Li, S.; Dai, W.; Lu, C.; Hu, W.; Zhang, L.; Du, J.; and Xiong, H. 2023. Learned Progressive Image Compression With Spatial Autoregression. In *2023 IEEE International Conference on Visual Communications and Image Processing (VCIP)*, 1–5.

Toderici, G.; O'Malley, S. M.; Hwang, S. J.; Vincent, D.; Minnen, D.; Baluja, S.; Covell, M.; and Sukthankar, R. 2016. Variable Rate Image Compression with Recurrent Neural Networks. In *International Conference on Learning Representations (ICLR)*.

Toderici, G.; Shi, W.; Timofte, R.; Theis, L.; Balle, J.; Agustsson, E.; Johnston, N.; and Mentzer, F. 2020. Workshop and challenge on learned image compression (clic2020). In *CVPR*.

Toderici, G.; Vincent, D.; Johnston, N.; Hwang, S. J.; Minnen, D.; Shor, J.; and Covell, M. 2017. Full Resolution Image Compression with Recurrent Neural Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5435–5443.

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

Wang, L.; Lu, H.; Wang, Y.; Feng, M.; Wang, D.; Yin, B.; and Ruan, X. 2017. Learning to detect salient objects with image-level supervision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 136–145.

Wang, Z.; Simoncelli, E.; and Bovik, A. 2003. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, 1398–1402 Vol.2.

Wu, S.; Chen, Y.; Liu, D.; and He, Z. 2025. Conditional Latent Coding with Learnable Synthesized Reference for Deep Image Compression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 12, 12863–12871.

Yang, N.; Chen, S.; Zhang, H.; and Berry, R. 2025. Beyond the Edge: An Advanced Exploration of Reinforcement Learning for Mobile Edge Computing, Its Applications, and Future Research Trajectories. *IEEE Communications Surveys & Tutorials*, 27(1): 546–594.

Zeng, F.; Tang, H.; Shao, Y.; Chen, S.; Shao, L.; and Wang, Y. 2025. MambaIC: State Space Models for High-Performance Learned Image Compression. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 18041–18050.

Zheng, C.; Hong, X.; Ding, D.; Vargaftik, S.; Ben-Itzhak, Y.; and Zilberman, N. 2024. In-Network Machine Learning Using Programmable Network Devices: A Survey. *IEEE Communications Surveys & Tutorials*, 26(2): 1171–1200.

Zhu, X.; Song, J.; Gao, L.; Zheng, F.; and Shen, H. T. 2022. Unified multivariate gaussian mixture for efficient neural image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 17612–17621.