

# DynaQuant: Dynamic Mixed-Precision Quantization for Learned Image Compression

Youneng Bao<sup>1</sup>, Yulong Cheng<sup>2</sup>, Yiping Liu<sup>2</sup>, Yichen Yang<sup>3</sup>, Peng Qin<sup>4</sup>,  
Mu Li<sup>2,\*</sup>, Yongsheng Liang<sup>1,2,\*</sup>

<sup>1</sup>College of Electronics and Information Engineering, Shenzhen University

<sup>2</sup>Harbin Institute of Technology, Shenzhen

<sup>3</sup>Marine Design and Research Institute of China

<sup>4</sup>China Telecom Group Qinhuangdao Branch

baoyouneng@163.com, {23s151073, yipingliu}@stu.hit.edu.cn, yicen1994@163.com,  
2244626474@qq.com, {limu2022, liangys}@hit.edu.cn

## Abstract

Prevailing quantization techniques in Learned Image Compression (LIC) typically employ a static, uniform bit-width across all layers, failing to adapt to the highly diverse data distributions and sensitivity characteristics inherent in LIC models. This leads to a suboptimal trade-off between performance and efficiency. In this paper, we introduce DynaQuant, a novel framework for dynamic mixed-precision quantization that operates on two complementary levels. First, we propose content-aware quantization, where learnable scaling and offset parameters dynamically adapt to the statistical variations of latent features. This fine-grained adaptation is trained end-to-end using a novel Distance-aware Gradient Modulator (DGM), which provides a more informative learning signal than the standard Straight-Through Estimator. Second, we introduce a data-driven, dynamic bit-width selector that learns to assign an optimal bit precision to each layer, dynamically reconfiguring the network's precision profile based on the input data. Our fully dynamic approach offers substantial flexibility in balancing rate-distortion (R-D) performance and computational cost. Experiments demonstrate that DynaQuant achieves rd performance comparable to full-precision models while significantly reducing computational and storage requirements, thereby enabling the practical deployment of advanced LIC on diverse hardware platforms.

Code — <https://github.com/baoyu2020/DynaQuant>

## Introduction

Learned Image Compression (LIC) has emerged as a new paradigm in image coding, achieving remarkable success in recent years. By replacing traditional hand-crafted modules—such as transforms, quantization, and entropy coding—with end-to-end optimized deep neural networks, LIC models (Ballé et al. 2018; He et al. 2022; Bao et al. 2023; Jiang et al. 2023b; Zeng et al. 2025) have consistently surpassed the R-D performance of state-of-the-art conventional codecs like BPG (Sullivan et al. 2012) and VVC (Bross et al. 2021). This data-driven methodology offers unprecedented

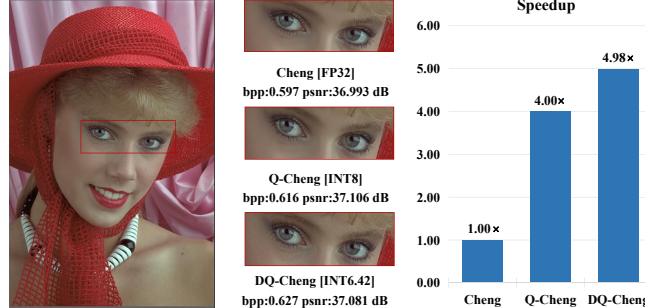


Figure 1: Visual and quantitative comparison of the DynaQuant method on the *kodim04*, using Cheng2020 (Cheng et al. 2020) as the baseline. Our proposed two quantization strategies—fixed bit-width quantization (Q-Cheng) and dynamic bit-width quantization (DQ-Cheng)—achieve comparable performance to the full-precision method while delivering approximately  $5 \times$  speedup.

flexibility and performance, marking a significant milestone in image coding.

Despite their impressive R-D performance, the practical deployment of these advanced LIC models is severely hampered by their substantial computational complexity and large memory footprint. The deep, complex architectures, while powerful, are computationally intensive, rendering them ill-suited for real-time applications on resource-constrained hardware such as smartphones, drones, and other edge devices.

To bridge this gap, model quantization has become a standard technique for compressing and accelerating neural networks. However, most existing quantization methods, often developed for robust high-level vision tasks like image classification, are suboptimal for the high-fidelity demands of image compression. These methods typically apply a static, uniform bit-width across all network layers. This “one-size-fits-all” approach fundamentally ignores two critical properties of LIC models: (1) Content-dependent dynamics: The statistical distributions of latent representations in LIC models are highly non-stationary and vary significantly with the

\*Corresponding author

Copyright © 2026, Association for the Advancement of Artificial Intelligence ([www.aaai.org](http://www.aaai.org)). All rights reserved.

content of each input image. A fixed set of quantization parameters cannot adapt to these dynamic distributions, leading to significant information loss. (2) Layer-wise sensitivity: Different layers within an LIC model exhibit vastly different sensitivities to quantization noise. Forcing a uniform precision profile is inefficient, as it may under-quantize robust layers while over-quantizing sensitive ones, resulting in a poor trade-off between accuracy and complexity.

To address these limitations, we propose DynaQuant, a framework for dynamic mixed-precision quantization tailored specifically for LIC. Our key insight is that an optimal quantization strategy must be dynamic at two complementary levels: the parameter level and the architectural level. First, to handle content-dependent data distributions, inspired by the soft to hard method (Agustsson et al. 2017), we introduce content-aware quantization, where the quantization parameters are learned and dynamically adjusted for each input. This process is guided by a novel distance-aware gradient modulator, which provides a more meaningful learning signal for the non-differentiable rounding operation than the conventional STE. Second, to address varying layer sensitivities, we design a lightweight, data-driven dynamic bit-width selector. This module learns to assign an optimal bit-width to each layer on-the-fly, creating an input-specific precision profile for the network architecture. This dual-level dynamic adaptation allows for an unprecedentedly fine-grained balance between rate-distortion performance and computational efficiency. Our main contributions can be summarized as follows:

- A novel dynamic mixed-precision quantization framework, DynaQuant, for LIC that adapts both quantization parameters and layer-wise bit-widths in a content-aware manner.
- A differentiable dynamic bit-width selector that learns to allocate optimal precision to each network layer by jointly optimizing the R-D loss of LIC with differentiable mixed-precision allocation.
- An extensive experiment demonstrating that DynaQuant achieves R-D performance on par with full-precision models while significantly reducing computational complexity and model size, thereby enabling practical deployment of LIC on diverse hardware platforms.

## Related Work

### Learned Image Compression

LIC has surpassed traditional codecs like JPEG (Wallace 1991), JPEG2000 (Skodras, Christopoulos, and Ebrahimi 2002) and VVC (Bross et al. 2021) in rate-distortion performance. This was pioneered by end-to-end trained autoencoder architectures (Ballé et al. 2018). Subsequent works have improved these architectural designs by introducing advanced network structures and attention mechanisms to enhance representation learning and model long-range dependencies (Tan et al. 2025; Bai et al. 2021; Zheng and Gao 2024). Recent methods leverage multi-modality networks with semantic priors (Jiang et al. 2023b) or replace CNNs with Transformer (Lu et al. 2021; Liu, Sun, and Katto 2023;

Li et al. 2023) and Mamba (Qin et al. 2024; Wu et al. 2025) for improved global modeling while preserving local detail (Liu, Sun, and Katto 2023; Zeng et al. 2025). In parallel, advances in entropy and context modeling have improved LIC performance, such as multi-reference entropy models for more accurate probability estimation (Jiang et al. 2023a; Bao et al. 2025). However, these collective advances in network architecture and entropy modeling substantially increase computational complexity, limiting practical deployment on resource-constrained devices.

To reduce the high computational cost of LIC, various model compression techniques have been explored. Pruning methods reduce model parameters via regularization-based sparsification (Chen and Yang 2023) or structured removal of network components (Kim et al. 2020; Luo et al. 2022). An alternative paradigm fits lightweight models to individual images, thereby simplifying the entropy model and feature extractors (Kim et al. 2024; Ballé et al. 2025; Zhang, Chen, and Liu 2025).

Parameter quantization is also widely used, employing strategies like layer-wise or low-bit quantization to reduce model size and accelerate inference (Hong et al. 2020; Jeon, Yu, and Lee 2023). Among these techniques, quantization is particularly suitable for practical deployment due to its implementation simplicity and hardware support.

However, existing quantization methods in LIC are typically static. They often apply uniform bit-widths and treat model components in isolation, an approach that lacks joint dynamic optimization. Consequently, these methods yield suboptimal R-D trade-offs and fail to fully exploit the potential of quantization.

### Model Quantization

Model quantization is typically divided into two categories: Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT).

**PTQ** quantizes models post-training without retraining but often suffers severe performance degradation at ultra-low bit-widths. Various methods have been proposed to mitigate this issue: data-adaptive rounding via quadratic optimization and continuous relaxation (Nagel et al. 2020), random dropping of quantization during inference to maintain accuracy (Wei et al. 2022), weight quantization guided by second-order information (Frantar, Singh, and Alistarh 2023), and direct optimization of the rate-distortion loss for task-oriented image compression (Shi, Lu, and Ma 2023).

**QAT** integrates quantization into the training loop, learning parameters via backpropagation to mitigate accuracy loss. Its development centers on two core challenges: devising adaptive quantization strategies and ensuring training stability. Static parameters struggle with data and layer diversity, while quantization's discrete nature destabilizes gradients. Solutions include learnable step sizes for data adaptation (Esser et al. 2019), quantization interval learning for stability (Jung et al. 2018), and content-adaptive methods that dynamically adjust parameters to each input (Liu et al. 2022).

## Mixed Precision Quantization

Mixed-precision quantization assigns different bit-widths to different layers, mitigating the performance loss of uniform-precision methods. This allocation can be modeled as a search problem, solved using reinforcement learning (Wang et al. 2019) or search-based methods at kernel-level granularity (Lou et al. 2019). Alternatively, bit-widths can be determined based on layer sensitivity, with some methods estimating this sensitivity through second-order Hessian information (Dong et al. 2019a,b). Other works (Liu, Simonyan, and Yang 2018) formulate bit-width allocation as a differentiable optimization problem, enabling layer precisions to be learned directly via gradient descent. In LIC, some approaches guide bit-width allocation by analyzing feature entropy (Sun et al. 2022). Others (Hossain, Duan, and Zhu 2024) have proposed flexible mixed-precision frameworks that provide customizable solutions for LIC models.

To address the limitation of employing a static, uniform bit-width across all layers, we propose a bit-selector with key advantages: (1) minimal computational overhead; (2) direct rate-distortion optimization. To our knowledge, this is the first end-to-end image compression method that combines trainable quantization parameters with differentiable mixed-precision allocation.

## Proposed Method

### Preliminaries: Quantization-Aware Training

QAT simulates the effects of quantization during training, allowing the model to adapt to precision loss. For a given full-precision input  $x$ , a  $b$ -bit asymmetric quantizer first maps it to an integer  $x_q$  using a scale factor  $s$  and a zero-point  $z$ :

$$x_q = \text{round} \left( \text{clip} \left( \frac{x}{s} + z, n_{\min}, n_{\max} \right) \right), \quad (1)$$

where  $[n_{\min}, n_{\max}]$  defines the quantization range (e.g.,  $[0, 2^b - 1]$  for unsigned  $b$ -bit integers). The scale  $s$  and zero-point  $z$  are typically pre-computed from the activation statistics. The de-quantized value  $\tilde{x}$ , which approximates the original input  $x$ , is recovered as:

$$\tilde{x} = s \cdot (x_q - z). \quad (2)$$

In conventional QAT, the scale  $s$  and zero-point  $z$  are statically computed from aggregated activation statistics and remain fixed during the inference stage.

To enable gradient-based optimization, the non-differentiable *round* function is commonly handled by the STE, which approximates its gradient as an identity function:

$$\frac{\partial \mathcal{L}}{\partial x} \approx \frac{\partial \mathcal{L}}{\partial \tilde{x}} \quad \text{if } x \in [s(n_{\min} - z), s(n_{\max} - z)]. \quad (3)$$

However, these two cornerstones of standard QAT—static parameters and a coarse, uniform gradient—are fundamentally ill-suited for the high-fidelity demands of LIC. Static parameters can not adapt to content-dependent variations in latent distributions, while the STE’s uniform gradient signal fails to provide meaningful feedback for optimizing the quantization process itself.

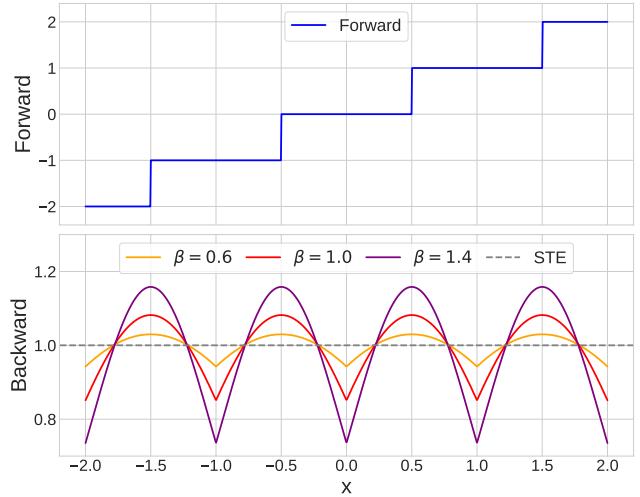


Figure 2: Gradient proxy function (top) and its derivative (bottom). The derivative exhibits periodic oscillations, reaching minima at  $x=0$  and  $x=1$  with peaks at  $x=0.5$ . All values remain strictly positive, and the amplitude is modulated by  $\beta$ , ensuring adaptive gradient scaling.

### Dynamic Parameter Adaptation (DPA)

To overcome the rigidity of static quantization, we introduce a mechanism that dynamically adapts quantization parameters to the specific characteristics of each input, combining content-aware parameter learning with a more principled gradient approximation.

**Content-Aware Quantization Mapping** The latent representations in LIC models exhibit highly dynamic and non-uniform distributions that depend on image content. To adapt to these distributions, we eschew fixed quantization parameters. Instead, we define the scale factor  $s$  and zero-point  $z$  as learnable, per-channel parameters. This allows the network to learn an optimal quantization mapping for each feature map, minimizing rate-distortion loss. The forward pass follows the standard asymmetric quantization-dequantization process:

$$\tilde{x} = s \cdot \left( \text{round} \left( \text{clip} \left( \frac{x}{s} + z, n_{\min}, n_{\max} \right) \right) - z \right), \quad (4)$$

where the clipping range  $[n_{\min}, n_{\max}]$  is set to  $[0, 2^b - 1]$  for a  $b$ -bit quantizer. The main challenge, however, lies in optimizing  $s$  and  $z$  effectively, which requires a valid gradient through the non-differentiable rounding function.

**Distance-Aware Gradient Modulation (DGM)** The Straight-Through Estimator, which approximates the rounding gradient as a constant 1, is the de facto standard. However, it implicitly assumes that quantization error is uniformly sensitive to changes in the input, an assumption that does not hold. The error is most sensitive near the quantization decision boundaries (i.e., half-integer values like 0.5).

To provide a more informative learning signal, we introduce a Distance-Aware Gradient Modulation (DGM) mechanism. The core principle of DGM is to make the magni-

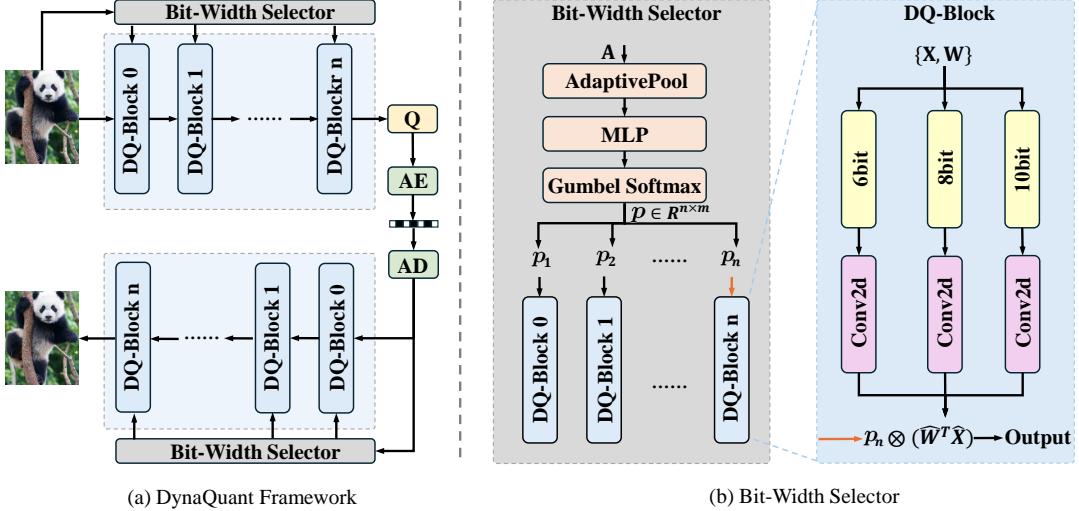


Figure 3: (a) DynaQuant Framework Overview. DQ-Block is the dynamic quantization block, and Bit-Width Selector is the bit-width selector that dynamically allocates quantization precision for each layer. (b) Bit-Width Selector and DQ-Block Structure. The bit-width selector processes input activation  $A$  through adaptive pooling, MLP, and Gumbel Softmax to output bit-width selection probability distribution  $p_1, p_2, \dots, p_n$ . DQ-Block quantizes the input  $\{X\}$  and learnable parameters  $\{W\}$  within the module according to the corresponding bit-widths based on the probability distribution, and finally generates the output through probability-weighted fusion.

tude of the backpropagated gradient a function of the input's proximity to the nearest decision boundary. Inspired by the work (Qin et al. 2023) on model parameter quantization, we define a new gradient proxy  $g'(x)$  to implement DGM:

$$\frac{\partial \text{round}(x)}{\partial x} \triangleq g'(x) = f(\text{dist}(x, \text{boundary})). \quad (5)$$

Specifically, we formulate  $g(x)$  as:

$$g(x) = \frac{1}{2} \cdot \frac{\tanh(\beta(x - \lfloor x \rfloor) - 0.5)}{\tanh(0.5)} + 0.5 \quad (6)$$

where  $x$  is the value before rounding and  $\beta$  is a shape factor. As illustrated in Fig. 2, this function yields a gradient that gradually decays as the input moves from decision boundaries toward quantization centers. Consequently, parameters associated with uncertain, boundary-adjacent values receive larger updates, while those associated with stable values receive smaller updates. This targeted gradient modulation encourages the model to learn better quantization parameters, enabling content-adaptive quantization and improving overall quantization performance.

### Dynamic Bit-Width Selector (DBWS)

Beyond adapting parameters, we introduce a method for layer-wise adaptive bit allocation. Instead of relying on costly search algorithms, we design a lightweight, data-driven module that learns to assign an optimal bit-width to each layer in an end-to-end fashion.

**Content-Aware Bit-Width Selector** We propose a differentiable bit-width selector, a small network module that predicts the optimal bit-width for a given layer based on its activation statistics. The selector takes the input activation tensor  $A \in \mathbb{R}^{C \times H \times W}$  and performs the following three steps:

1. **Global Feature Extraction:** An adaptive pooling layer (e.g., AdaptivePool) first condenses the spatial dimensions of the activation tensor into a fixed-size feature vector  $\mathbf{h}_{\text{pool}} = \text{AdaptivePool}(A)$ . This vector captures the global statistical properties of the activations while remaining agnostic to the input tensor's dimensions.

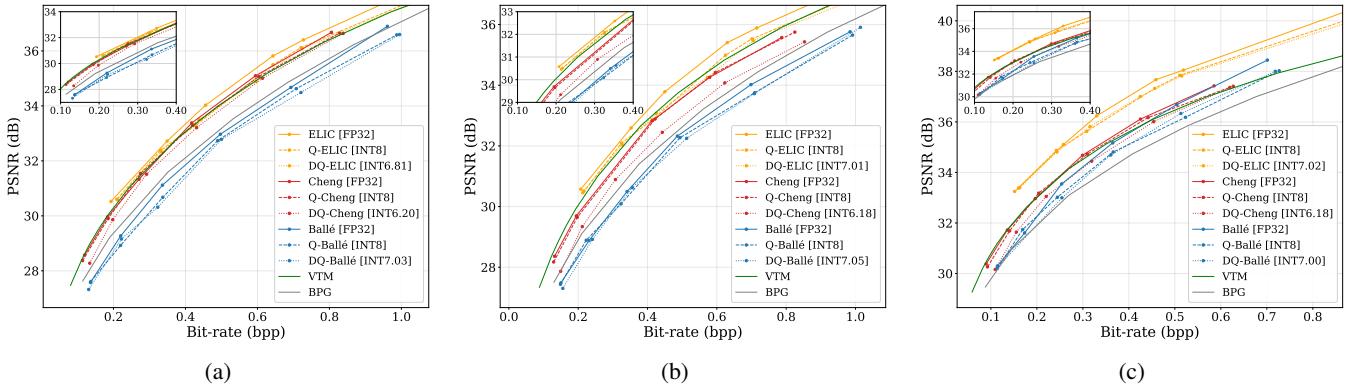
2. **Bit-Width Prediction:** The feature vector is then processed by a two-layer MLP to predict a probability distribution over a set of candidate bit-widths  $\mathcal{B} = \{b_1, b_2, \dots, b_M\}$ .

$$\mathbf{p} = \text{Softmax}(\text{MLP}(\mathbf{h}_{\text{pool}})) \quad (7)$$

where  $\mathbf{p} = \{p_1, p_2, \dots, p_N\}$ .  $N$  is the total number of quantizable blocks, for each quantizable block  $l$ .  $\mathbf{p}_l \in \mathbb{R}^M$  is the probability vector, with  $(\mathbf{p}_l)_k$  representing the probability of selecting bit-width  $b_k$ .

3. **Stochastic Selection for Training:** To make the discrete selection process differentiable, we employ the Gumbel-Softmax reparameterization trick during training. This allows us to sample a one-hot vector  $\mathbf{p}_l$  from the categorical distribution defined by  $\mathbf{p}$  while allowing gradients to flow back to the selector's parameters. The effective bit-width for the layer is then calculated as  $b_l = \sum_{k=1}^M (\mathbf{p}_l)_k \cdot b_k$ .

During inference, we deterministically select the bit-width with the highest probability to ensure stable and efficient execution:  $b_l^* = b_k$  where  $k = \text{argmax}(\mathbf{p}_l)$ . This mechanism allows each layer to dynamically choose its precision.



**Figure 4: R-D Performance.**(a) Kodak; (b) JPEG-AI; (c) CLIC. Quantization schemes: [FP32] for original 32-bit float, [INT8] for uniform 8-bit quantization, and [INTX.YY] for our mixed-precision quantization where X.YY indicates the average bit-width (e.g., 6.81-bit for DQ-ELIC, 6.20-bit for DQ-Cheng). “Q-ELIC” refers to applying DPA to each layer of ELIC, while “DQ-ELIC” denotes the application of a DBWS to each layer of ELIC. Best viewed in color.

## Joint Optimization Framework

Our framework is trained end-to-end by minimizing a composite loss function that jointly optimizes for rate-distortion performance and model complexity. The overall learning objective  $\mathcal{L}$  is defined as:

$$\mathcal{L} = R + \lambda D + \gamma \mathcal{L}_{\text{bits}}, \quad (8)$$

where  $R$  is the estimated bitrate of the quantized latent representation, derived from the entropy model.  $D$  measures the distortion between the original image and the reconstructed image (e.g., using MSE or MS-SSIM);  $\mathcal{L}_{\text{bits}}$  is a complexity regularization term, and  $\lambda$  is a hyperparameter that balances the two objectives. The hyperparameter  $\gamma$  is a hyperparameter that balances the R-D performance and bit-width.

The core of our adaptive training is the complexity regularization term,  $\mathcal{L}_{\text{bits}}$ , which guides the bit-width selector. To encourage the network to favor lower-precision configurations, we define this term as the expected average bit-width across all  $L$  dynamically quantized layers:

$$\mathcal{L}_{\text{bits}} = \frac{1}{L} \sum_{l=1}^L \mathbb{E}_{\mathbf{p}_l}[b] = \frac{1}{L} \sum_{l=1}^L \sum_{k=1}^M (\mathbf{p}_l)_k \cdot b_k, \quad (9)$$

where  $(\mathbf{p}_l)_k$  is the probability of selecting the  $k$ -th candidate bit-width  $b_k \in \mathcal{B}$  for layer  $l$ . This loss term creates a direct optimization pressure on the bit-width selection policy, rewarding it for reducing the average bit-width. By adjusting  $\gamma$ , we can effectively navigate the trade-off between model performance and its computational/storage footprint, producing models that are optimized for various efficiency constraints.

## Experiments

### Experimental Setup

**Datasets and Evaluation Metrics** We evaluate our method on three datasets: Kodak (Franzen 1999), JPEG-AI (JPEG Committee 2020) and the CLIC (Toderici et al.

2020). Image quality is assessed using PSNR, while compression efficiency is measured in bits per pixel (bpp). BD-Rate loss is reported to quantify the average BPP difference for the same quality.

**Baseline LIC Model and Comparison Methods** We adopt the Cheng2020(Cheng et al. 2020), Ballé (Ballé et al. 2018), and ELIC (He et al. 2022) architectures as our baseline full-precision (FP32) LIC models, recognized for their robust performance. Our proposed DynaQuant method is compared against their full-precision versions, as well as several quantization methods: 8-bit Fixed Precision Quantization (FPQ), FMPQ (Faisal Hossain, Duan, and Zhu 2024), RAQ (Hong et al. 2020), and RDO-PTQ (Shi, Lu, and Ma 2023). Implementation details are provided in the Appendix.

## Main Results

**Rate-Distortion Performance** Fig. 4 shows R-D curves across datasets and LIC models, where our proposed two quantization strategies closely match full-precision baselines with minimal degradation over 0.1–1.0 bpp. Table 1 reports fixed-bit-width quantization achieving near-optimal BD-Rate losses (Q-Cheng: 1.60%, Q-Ballé: 5.01%) compared to FMPQ benchmarks (1.30%, 7.50%). Dynamic bit-width quantization improves speedup ( $4.61\times$ ,  $5.17\times$ ,  $4.55\times$  across models, surpassing  $4\times$ ) with slightly higher BD-Rate loss. These results confirm that DynaQuant’s content-aware quantization and dynamic bit-width optimization enable efficient LIC deployment on resource-constrained devices.

### Ablation Studies

**General Ablation** Table 2 presents the results of our general ablation study. **First**, our DPA method outperforms the classical 8-bit PAMS quantization, achieving an approximate bpp of 0.83 (ranging from 0.827 to 0.828), the highest PSNR (36.649), and the lowest R-D loss (1.56), confirming the effectiveness of our fixed-bit-width quantization approach. **Second**, integrating our dynamic bit-width method (DQ) with DPA reduces the average bitwidth from 8 to 6.42

Table 1: BD-Rate loss (%) and computational speed up ( $\times$ ) relative to baseline 32-bit full precision models on three standard datasets: Kodak, JPEG-AI, and CLIC. “Q-ELIC” refers to applying Dynamic Parameter Adaptation (DPA) to each layer of ELIC, while “DQ-ELIC” denotes the application of a Dynamic Bit-Width Selector (DBWS) to each layer of ELIC. A dash “–” denotes unreported results; the asterisk “\*” indicates our calculated results.

Model	Method	Kodak		JPEG-AI		CLIC		Average			
		BD-Rate	Speedup	BD-Rate	Speedup	BD-Rate	Speedup	Bitwidth	BD-Rate	Speedup	Model Size
ELIC	Full precision	0.00	1.00 $\times$	0.00	1.00 $\times$	0.00	1.00 $\times$	32.00	0.00	1.00 $\times$	137.11 MB
	Q-ELIC	5.97	4.00 $\times$	6.24	4.00 $\times$	2.55	4.00 $\times$	8.00	4.92	4.00 $\times$	34.28 MB
	DQ-ELIC	7.62	4.70 $\times$	7.53	4.56 $\times$	4.01	4.56 $\times$	6.95	6.39	4.61 $\times$	29.78 MB
Cheng	Full precision	0.00	1.00 $\times$	0.00	1.00 $\times$	0.00	1.00 $\times$	32.00	0.00	1.00 $\times$	45.08 MB
	8-bit FPQ	2.05	*3.98 $\times$	–	–	3.54	*3.98 $\times$	8.00	2.80	*3.98 $\times$	*11.31 MB
	FMPQ	0.89	4.00 $\times$	–	–	1.70	4.00 $\times$	–	1.30	4.00 $\times$	*11.27 MB
	RAQ	27.84	–	–	–	–	–	–	27.84	–	–
	RDO-PTQ	4.88	4.00 $\times$	–	–	–	–	8.00	4.88	4.00 $\times$	*11.27 MB
	Q-Cheng	1.02	4.00 $\times$	1.18	4.00 $\times$	2.61	4.00 $\times$	8.00	1.60	4.00 $\times$	11.27 MB
	DQ-Cheng	7.15	5.16 $\times$	16.52	5.17 $\times$	12.87	5.18 $\times$	6.19	12.18	5.17 $\times$	8.72 MB
Ballé	Full precision	0.00	1.00 $\times$	0.00	1.00 $\times$	0.00	1.00 $\times$	32.00	0.00	1.00 $\times$	19.37 MB
	8-bit FPQ	7.44	*3.97 $\times$	–	–	8.95	*3.97 $\times$	8.00	8.20	*3.97 $\times$	*4.88 MB
	FMPQ	6.48	*3.98 $\times$	–	–	8.95	*3.98 $\times$	–	7.50	*3.98 $\times$	*4.87 MB
	Q-Ballé	5.85	4.00 $\times$	2.59	4.00 $\times$	6.60	4.00 $\times$	8.00	5.01	4.00 $\times$	4.84 MB
	DQ-Ballé	7.63	4.55 $\times$	4.84	4.54 $\times$	8.06	4.57 $\times$	7.03	6.84	4.55 $\times$	4.26 MB

Table 2: General ablation study of our two quantization strategies: DPA (fixed bit-width, INT8/INT6) and DPA-DQ (dynamic bit-width quantization, DBWS), compared to Vanilla and PAMS quantization.

Method	Bitwidth	bpp	PSNR	R-D loss
Vanilla	32	0.831	36.91	1.52
<i>Fixed Bit-width Methods</i>				
+PAMS	8	0.83	36.185	1.64
+DPA[INT8] (Ours)	8	0.828	36.649	<b>1.56</b>
+DPA[INT6] (Ours)	6	0.827	35.664	1.74
<i>Dynamic bit-width Methods</i>				
+PAMS-DQ	6.85	0.892	30.262	4.28
+DPA-DQ (Ours)	6.42	0.838	36.636	<b>1.57</b>
+DPA-DQ (Ours)*	6.02	0.882	36.231	1.68

while maintaining a competitive PSNR (36.636) and R-D loss (1.57), underscoring the efficiency of dynamic optimization. **Third**, the combination of DPA and DQ demonstrates a synergistic effect, outperforming the sum of individual contributions (i.e., DPA-DQ exceeds PAMS-DQ and simple DPA with INT6), with an optimized bitwidth of 6.02, a PSNR of 36.231, and an R-D loss of 1.68. This suggests that the integrated approach enhances performance beyond additive gains, validating the efficacy of each component and the superior impact of their combined use.

**Ablation of Dynamic Parameter Adaptation** Table 3 shows that removing any DPA component degrades performance (e.g., excluding  $s$  reduces PSNR from 36.649

Table 3: Ablation study of our proposed fixed bit-width quantization strategy, DPA.

$s$	$z$	$g(x)$	Bitwidth	bpp	PSNR	R-D Loss
✓	✓	✓	8	0.828	<b>36.649</b>	<b>1.56</b>
✗	✓	✓	8	0.83	36.185	1.65
✓	✗	✓	8	0.824	36.323	1.58
✓	✓	✗	8	0.842	36.288	1.63

Table 4: Ablation study of our proposed dynamic bit-width quantization strategy (DBWS) with different bitwidth choices.

Bitwidth Choice	Avg Bitwidth	bpp	PSNR	R-D Loss
{4,6,8}	5.47	0.866	36.432	1.64
{6,8,10}	6.42	0.838	36.636	1.57

to 36.185 and increases R-D loss from 1.56 to 1.65). These results highlight the importance of each component—learnable  $s$ ,  $z$ , and  $g(x)$ —in improving quantization accuracy and preserving compression quality.

**Ablation of Dynamic Bit-Width Selector** Table 4 shows that the dynamic bit-width strategy (DBWS) remains effective across different bit-width sets. The {6, 8, 10} set offers a better trade-off between efficiency and fidelity, while {4, 6, 8} achieves lower average bit-width at the cost of reduced PSNR.

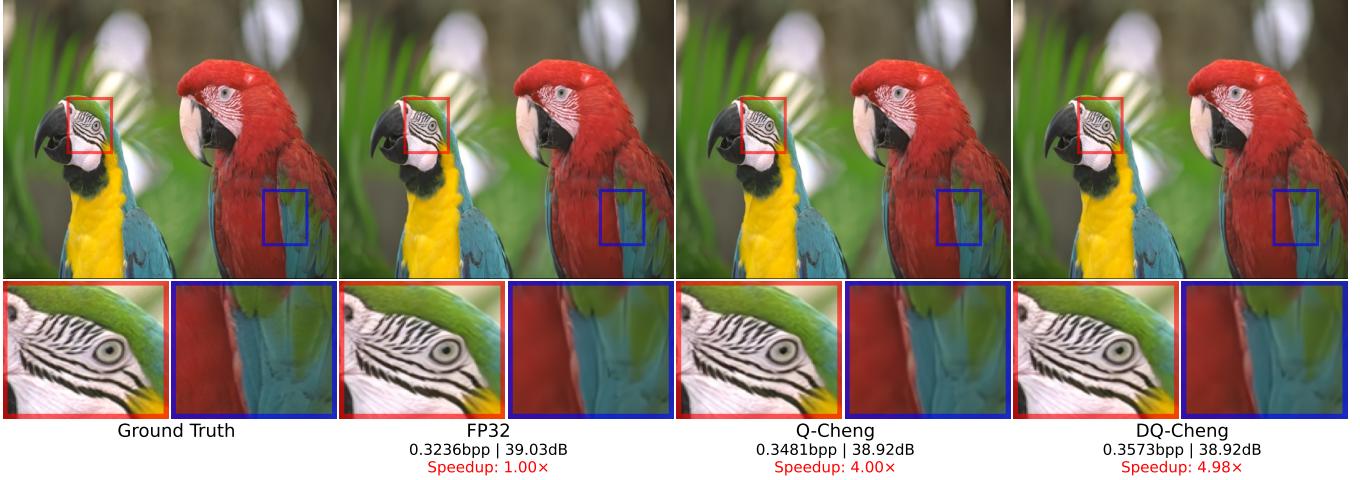


Figure 5: Qualitative comparison on a Kodak image (e.g., “*kodim23*”) using the base model Cheng2020. (a) Full-size images (first row). (b) Zoomed-in regions highlighting texture/edge details (second row). Methods: From left to right—Ground Truth, 32-bit full-precision, Application of our Dynamic Parameter Adaptation (Q-Cheng), and application of a Dynamic Bit-Width Selector (DQ-Cheng). Metrics include bpp, PSNR, and speedup. Best viewed digitally and zoomed in.

## Qualitative Results

**Visualization Results** Fig. 5 demonstrates that our proposed quantization strategies (e.g., Q-Cheng and DQ-Cheng) achieve significant speedups of  $4.00\times$  and  $4.98\times$  respectively, despite slight bpp increases, while maintaining near 39 dB visual quality comparable to the 32-bit full-precision baseline.

**Bit-width Selection Results** Fig. 6 showcases dynamic bit-width allocation across images and layers. For example, *Kodim14* with more detailed texture assigns a 10-bit width to its gs-1 layer, exceeding the 8-bit allocation in other images. Edge layers (e.g., ga-0, ga-6, gs-1) exhibit higher precision than intermediate layers, indicating bottleneck layers require increased bit-width. These findings confirm that our Dynamic Bit-width Quantization adapts precision to data complexity and network structure.

## Conclusion

In this work, we presented DynaQuant, a dynamic mixed-precision quantization framework for LIC that jointly adapts quantization parameters and layer-wise bit-widths in a content-aware manner. By employing a distance-aware gradient modulator to train learnable quantization parameters, the method provides more informative gradients and mitigates performance degradation. A differentiable dynamic bit-width selector further learns to allocate optimal precision to each network layer by jointly optimizing the rate-distortion loss with a differentiable mixed-precision allocation cost. Extensive experiments show that DynaQuant achieves R-D performance close to that of full-precision models while significantly reducing computational complexity and model size, yielding speedups of up to  $5.17\times$ .

Although a performance trade-off remains at maximum acceleration, future work may refine this balance by incorpo-

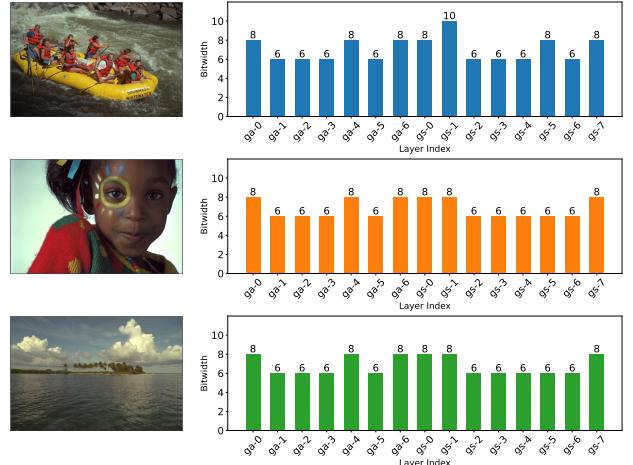


Figure 6: Learned bit-width allocation across network layers based on Cheng2020. Left: Kodim14/15/16 test images (top to bottom). Right: Optimized bit-widths automatically assigned to each layer.

rating hardware-specific latency models into the optimization objective.

## Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grants 62571160 and 62472124), the Shenzhen Colleges and Universities Stable Support Program (Grant GXWD20220811170130002), the Engineering Technology R&D Center of Guangdong Provincial Universities (Grant 2024GCZX004), and the Major Project of Guangdong Basic and Applied Basic Research (Grant 2023B0303000010).

## References

- Agustsson, E.; Mentzer, F.; Tschannen, M.; Cavigelli, L.; Timofte, R.; Benini, L.; and Gool, L. V. 2017. Soft-to-Hard Vector Quantization for End-to-End Learning Compressible Representations. In *NIPS*, 1141–1151.
- Bai, Y.; Yang, X.; Liu, X.; Jiang, J.; Wang, Y.; Ji, X.; and Gao, W. 2021. Towards End-to-End Image Compression and Analysis with Transformers. *arXiv:2112.09300*.
- Ballé, J.; Minnen, D.; Singh, S.; Hwang, S. J.; and Johnston, N. 2018. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*.
- Ballé, J.; Versari, L.; Dupont, E.; Kim, H.; and Bauer, M. 2025. Good, cheap, and fast: Overfitted image compression with Wasserstein distortion. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 23259–23268.
- Bao, Y.; Meng, F.; Li, C.; Ma, S.; Tian, Y.; and Liang, Y. 2023. Nonlinear Transforms in Learned Image Compression From a Communication Perspective. *IEEE Trans. Circuits Syst. Video Technol.*, 33(4): 1922–1936.
- Bao, Y.; Tan, W.; Jia, C.; Li, M.; Liang, Y.; and Tian, Y. 2025. ShiftLIC: Lightweight Learned Image Compression With Spatial-Channel Shift Operations. *IEEE Trans. Circuits Syst. Video Technol.*, 35(9): 9428–9442.
- Bross, B.; Wang, Y.-K.; Ye, Y.; Liu, S.; Chen, J.; Sullivan, G. J.; and Ohm, J.-R. 2021. Overview of the versatile video coding (VVC) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(10): 3736–3764.
- Chen, W.; and Yang, Q. 2023. Efficient pruning method for learned lossy image compression models based on side information. In *2023 IEEE International Conference on Image Processing (ICIP)*, 3464–3468. IEEE.
- Cheng, Z.; Sun, H.; Takeuchi, M.; and Katto, J. 2020. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7936–7945.
- Dong, Z.; Yao, Z.; Cai, Y.; Arfeen, D.; Gholami, A.; Mahoney, M. W.; and Keutzer, K. 2019a. HAWQ-V2: Hessian Aware trace-Weighted Quantization of Neural Networks. *CoRR*, abs/1911.03852.
- Dong, Z.; Yao, Z.; Gholami, A.; Mahoney, M.; and Keutzer, K. 2019b. HAWQ: Hessian AWare Quantization of Neural Networks with Mixed-Precision. *arXiv:1905.03696*.
- Esser, S. K.; McKinstry, J. L.; Bablani, D.; Appuswamy, R.; and Modha, D. S. 2019. Learned step size quantization. *arXiv preprint arXiv:1902.08153*.
- Faisal Hossain, M. A.; Duan, Z.; and Zhu, F. 2024. Flexible Mixed Precision Quantization for Learne Image Compression. In *2024 IEEE International Conference on Multimedia and Expo (ICME)*, 1–8.
- Frantar, E.; Singh, S. P.; and Alistarh, D. 2023. Optimal Brain Compression: A Framework for Accurate Post-Training Quantization and Pruning. *arXiv:2208.11580*.
- Franzen, R. 1999. Kodak Lossless True Color Image Suite. <http://r0k.us/graphics/kodak/>.
- 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.
- Hong, W.; Chen, T.; Lu, M.; Pu, S.; and Ma, Z. 2020. Efficient neural image decoding via fixed-point inference. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(9): 3618–3630.
- Hossain, M. A. F.; Duan, Z.; and Zhu, F. 2024. Flexible Mixed Precision Quantization for Learne Image Compression. In *2024 IEEE International Conference on Multimedia and Expo (ICME)*, 1–8. IEEE.
- Jeon, G.-W.; Yu, S.; and Lee, J.-S. 2023. Integer quantized learned image compression. In *2023 IEEE International Conference on Image Processing (ICIP)*, 2755–2759. IEEE.
- Jiang, W.; Yang, J.; Zhai, Y.; Ning, P.; Gao, F.; and Wang, R. 2023a. Mlic: Multi-reference entropy model for learned image compression. In *Proceedings of the 31st ACM International Conference on Multimedia*, 7618–7627.
- Jiang, X.; Tan, W.; Tan, T.; Yan, B.; and Shen, L. 2023b. Multi-modality deep network for extreme learned image compression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 1033–1041.
- JPEG Committee. 2020. JPEG AI Dataset. <https://jpeg.org/jpegai/dataset.html>. Training/validation dataset: 5264/350 images. Accessed: November 12, 2026.
- Jung, S.; Son, C.; Lee, S.; Son, J.; Kwak, Y.; Han, J.; and Choi, C. 2018. Joint Training of Low-Precision Neural Network with Quantization Interval Parameters. *CoRR*, abs/1808.05779.
- Kim, H.; Bauer, M.; Theis, L.; Schwarz, J. R.; and Dupont, E. 2024. C3: High-performance and low-complexity neural compression from a single image or video. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9347–9358.
- Kim, J.-H.; Choi, J.-H.; Chang, J.; and Lee, J.-S. 2020. Efficient deep learning-based lossy image compression via asymmetric autoencoder and pruning. In *Icassp 2020-2020 ieee international conference on acoustics, speech and signal processing (icassp)*, 2063–2067. IEEE.
- Li, H.; Li, S.; Dai, W.; Li, C.; Zou, J.; and Xiong, H. 2023. Frequency-aware transformer for learned image compression. *arXiv preprint arXiv:2310.16387*.
- Liu, H.; Simonyan, K.; and Yang, Y. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.
- 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, Z.; Wang, Y.; Han, K.; Ma, S.; and Gao, W. 2022. Instance-aware dynamic neural network quantization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 12434–12443.
- Lou, Q.; Liu, L.; Kim, M.; and Jiang, L. 2019. AutoQB: AutoML for Network Quantization and Binarization on Mobile Devices. *CoRR*, abs/1902.05690.
- Lu, M.; Guo, P.; Shi, H.; Cao, C.; and Ma, Z. 2021. Transformer-based image compression. *arXiv preprint arXiv:2111.06707*.
- Luo, A.; Sun, H.; Liu, J.; and Katto, J. 2022. Memory-efficient learned image compression with pruned hyperprior module. In *2022 IEEE International Conference on Image Processing (ICIP)*, 3061–3065. IEEE.
- Nagel, M.; Amjad, R. A.; Van Baalen, M.; Louizos, C.; and Blankevoort, T. 2020. Up or down? adaptive rounding for post-training quantization. In *International conference on machine learning*, 7197–7206. PMLR.

- Qin, H.; Zhang, Y.; Ding, Y.; Liu, X.; Danelljan, M.; Yu, F.; et al. 2023. QuantSR: accurate low-bit quantization for efficient image super-resolution. *Advances in neural information processing systems*, 36: 56838–56848.
- Qin, S.; Wang, J.; Zhou, Y.; Chen, B.; Luo, T.; An, B.; Dai, T.; Xia, S.; and Wang, Y. 2024. Mambavc: Learned visual compression with selective state spaces. *arXiv preprint arXiv:2405.15413*.
- Shi, J.; Lu, M.; and Ma, Z. 2023. Rate-distortion optimized post-training quantization for learned image compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 34(5): 3082–3095.
- Skodras, A.; Christopoulos, C.; and Ebrahimi, T. 2002. The JPEG 2000 still image compression standard. *IEEE Signal processing magazine*, 18(5): 36–58.
- 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.
- Sun, Z.; Ge, C.; Wang, J.; Lin, M.; Chen, H.; Li, H.; and Sun, X. 2022. Entropy-driven mixed-precision quantization for deep network design. *Advances in Neural Information Processing Systems*, 35: 21508–21520.
- Tan, W.; Meng, F.; Li, C.; Bao, Y.; and Liang, Y. 2025. Exploring High Dimensional Feature Space With Channel-Spatial Nonlinear Transforms for Learned Image Compression. *CAAI Trans. Intell. Technol.*, 10(4): 1235–1253.
- 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).
- Wallace, G. K. 1991. The JPEG Still Picture Compression Standard. *Commun. ACM*, 34(4): 30–44.
- Wang, K.; Liu, Z.; Lin, Y.; Lin, J.; and Han, S. 2019. HAQ: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 8612–8620.
- Wei, X.; Gong, R.; Li, Y.; Liu, X.; and Yu, F. 2022. Qdrop: Randomly dropping quantization for extremely low-bit post-training quantization. *arXiv preprint arXiv:2203.05740*.
- Wu, Z.; Du, H.; Wang, S.; Lu, M.; Sun, H.; Guo, Y.; and Yu, X. 2025. CMamba: Learned Image Compression with State Space Models. *arXiv preprint arXiv:2502.04988*.
- 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.
- Zhang, Z.; Chen, Z.; and Liu, S. 2025. Fitted Neural Lossless Image Compression. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 23249–23258.
- Zheng, H.; and Gao, W. 2024. End-to-End RGB-D Image Compression via Exploiting Channel-Modality Redundancy. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(7): 7562–7570.

## Appendix

This appendix provides comprehensive technical details on the following aspects:

1. The architectural implementation of the Bit-Width Selector module, including detailed hyperparameter configurations of convolutional and pooling layers, network topology, and data flow propagation paths;
2. Visualization results of bit-width allocation strategies of the proposed Bit-Width Selector module across different neural network architectures, demonstrating the quantization bit-width selection distribution for each functional module.

### A.1 DBWS Architecture Details

**A.1.1 Encoder-Decoder DBWS Configuration** For the encoder and decoder, each is equipped with its own independent dynamic bit-width selector, but both adopt identical network architectures. This symmetric architectural design ensures that the encoder and decoder can generate consistent bit-width selection strategies, eliminating the need to transmit additional bit-width configuration information during image transmission and significantly reducing communication overhead. In the following descriptions, the number of quantized layers in the encoder and decoder is uniformly denoted as  $BL$ .

**A.1.2 DBWS Design** As illustrated in Figure A1, the DBWS module adopts a hierarchical architecture with parameter configurations detailed in Table A1. The module comprises six core components:

- (1) **Adaptive Pooling Layer:** pools input features  $A$  to a fixed size ( $5 \times 5$ ), ensuring generalizability for different input resolutions;
- (2) **Flatten Layer:** converting feature maps to one-dimensional vectors;
- (3) **Multi-Layer Perceptron (MLP):** consisting of two linear layers and one dropout layer, implementing feature transformation and dimensionality reduction from  $N \times 25$  to 128, then outputting  $\text{num\_bits} \times BL$  dimensional feature vectors;
- (4) **Reshape Layer:** reconstructing outputs to  $(B \times BL, \text{num\_bits})$  shape;
- (5) **Gumbel Softmax Layer:** performing differentiable discrete sampling with hard quantization strategy ( $\tau = 1$ , `hard=True`) to output definitive bit-width configurations;
- (6) **Final Reshape:** reshaping to  $(B, BL, \text{num\_bits})$  to provide one-hot encoded bit-width selections for  $BL$  quantization blocks.

The DBWS module achieves adaptive optimal bit-width selection based on input features through end-to-end learning, balancing compression efficiency and reconstruction quality.

**Notice:** In our experimental setup and training process, the dynamic bit-width selector is applied exclusively to the main encoder-decoder module, while the hyperencoder adopts a uniform 8-bit fixed quantization strategy. This design decision is based on the following experimental analysis: the hyperencoder module is characterized by simple structure and fewer parameters, making further quantization compression yield limited benefits, with negative returns observed in some cases; meanwhile, each layer of this module exhibits high sensitivity to quantization errors, where low bit-width quantization easily leads to significant performance degradation; furthermore, given the relatively small computational load of the hyperencoder itself, introducing complex dynamic quantization strategies would add unnecessary system overhead with limited performance improvement. Therefore, adopting a uniform 8-bit quantization strategy for the hyperencoder represents the optimal choice, ensuring both module performance stability and avoiding excessive system architecture complexity.

Table A1: Detailed Parameter Configuration of DBWS

Layer	Input Shape	Output Shape	Parameters	Key Settings
AdaptivePool	(B, N, H, W)	(B, N, 5, 5)	0	output_size=(5,5)
Linear	(B, N×25)	(B, 128)	(N×25+1)×128	in_feature=N×5×5,out_feature=128
Dropout	(B, 128)	(B, 128)	0	p=0.2
Linear	(B, 128)	(B×BL,num_bits)	(128+1)×(num_bits×BL)	in_feature=128,out_feature= BL×num_bits
GumbelSoftmax	(B×BL,num_bits)	(B×BL,num_bits)	0	tau=1,hard=True

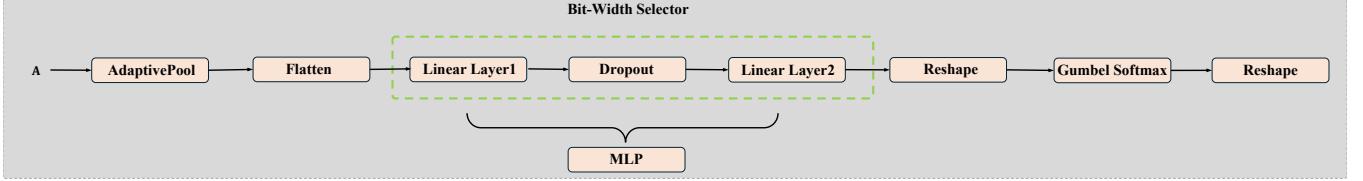


Figure A1: Detailed Architecture of DBWS

## A.2 DBWS Input Data Selection Strategy

In our experimental framework, the Dynamic Bit-Width Selector (DBWS) for the main encoder-decoder adopts a symmetric configuration design, where the encoder and decoder respectively utilize the output features from their corresponding first modules as inputs to their respective DBWS. The core design of this configuration strategy is as follows: First, the first processing modules of both encoder and decoder employ fixed 8-bit quantization to provide a stable feature foundation for subsequent bit-width selection; second, the encoder DBWS processes output features from the encoder's first module, while the decoder DBWS processes output features from the decoder's first module; finally, based on the respective DBWS outputs, all subsequent modules (2nd to  $BL$ th) in the encoder-decoder employ adaptive bit-width selection strategies. This symmetric configuration design maximizes quantization flexibility while ensuring computational efficiency, enabling most processing modules in the encoder-decoder to benefit from dynamic quantization advantages.

As illustrated in Figure A2, we present a comparative analysis of different layer outputs serving as DBWS inputs. After comprehensive trade-off analysis between dynamic quantization flexibility and computational complexity, we ultimately determine to adopt the output features from the respective first modules of the encoder-decoder as the corresponding DBWS input configuration.

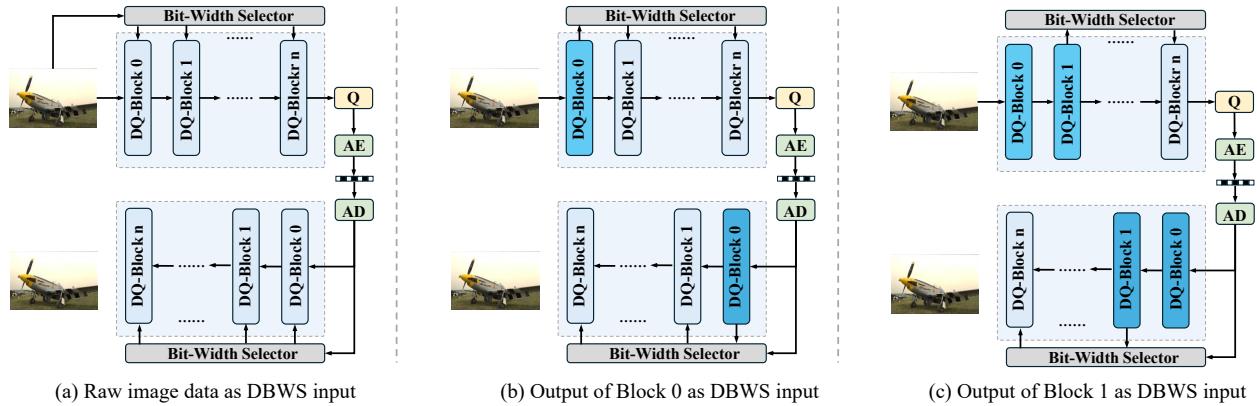


Figure A2: Dark blue DQ-Block modules denote fixed 8-bit quantization, while light blue modules represent adaptive quantization blocks.

## A.3 Cross-Architecture Results

To validate the generalizability of our Dynamic Bit-Width Selector (DBWS) Selector (DBWS), we extend the evaluation to two additional neural compression architectures: Ballé and ELIC. Figure A3 and Figure A4 presents the bit-width allocation patterns across different layers and test images for these alternative architectures.

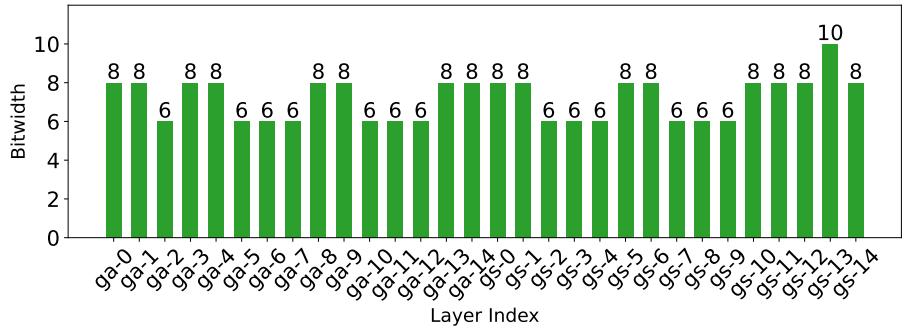
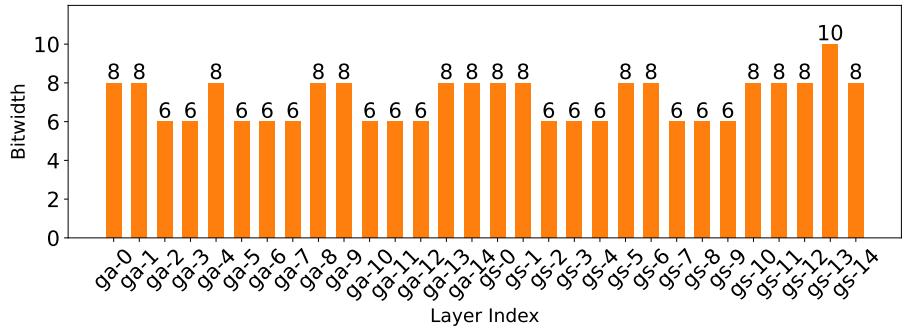
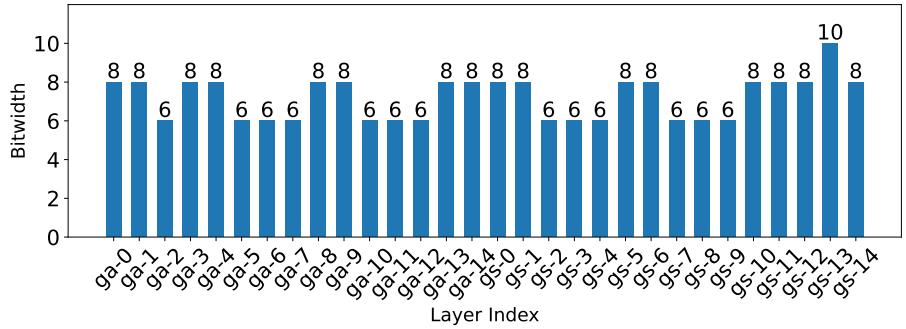


Figure A3: Learned bit-width allocation across network layers based on ELIC. Left: Kodim14/15/16 test images (top to bottom). Right: Optimized bit-widths automatically assigned to each layer.

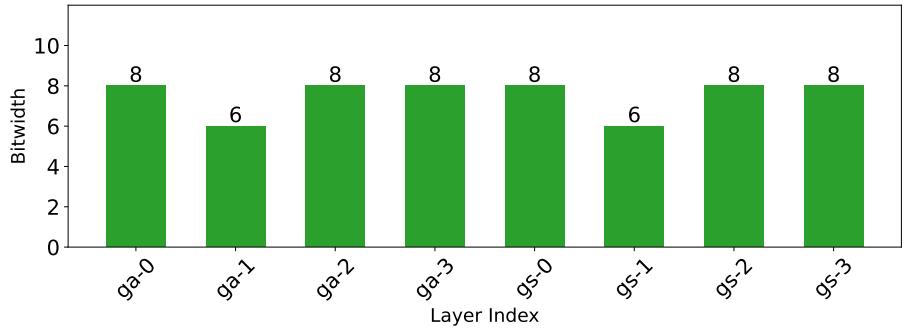
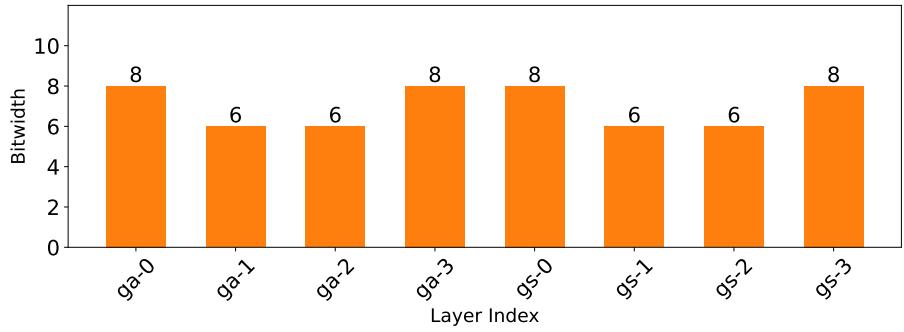
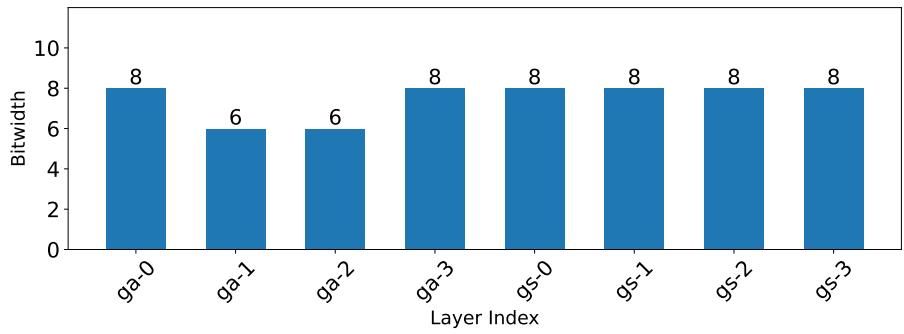


Figure A4: Learned bit-width allocation across network layers based on Ballé. Left: Kodim14/15/16 test images (top to bottom). Right: Optimized bit-widths automatically assigned to each layer.