WHERE SHOULD THE COMPUTE GO? PRACTICAL NEURAL IMAGE COMPRESSION ACROSS BUDGETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learned image compression has made substantial progress, yet its high computational cost remains a major barrier to practical deployment. Under tight complexity budgets, effective design requires not only reducing model complexity but also allocating computation across components. Taking edge-device complexity as the starting point, we present a systematic study of learned image compression under constrained budgets. Concretely, we first build a unified lightweight architecture and analyze computation allocation between the transform modules and the entropy model, then perform local structure search by independently optimizing and composing sub-modules, and finally construct a scalable model family via gradual expansion, with compact variants further improved by energy-based hierarchical pruning. The key aspect of our study is a systematic design procedure rather than any single network: under the unified architecture we establish, each specified complexity budget yields a configuration that consistently outperforms comparable low-complexity baselines across the evaluated constraints. As a representative result, with VTM-24.0 as the common anchor, our model achieves a 24.82% BD-rate reduction on Kodak relative to Shallow-NTC, a representative low-complexity decoding baseline, while reducing encoding and decoding complexity by about 84% and 13%, respectively.

1 INTRODUCTION

Learned image compression jointly optimizes transforms, quantization, and entropy models for rate-distortion performance (Ballé et al., 2017; 2021; Cheng et al., 2020). Recent methods improve compression through stronger transforms and context models (He et al., 2022; Jiang et al., 2025; Li et al., 2024; 2025). These improvements yield significant performance gains when the complexity budget is sufficient, allowing each module ample capacity and flexible design. However, under tighter budgets, increasing the capacity of one module inevitably comes at the expense of another. Therefore, how to allocate computational resources under a tight budget remains a compelling research problem.

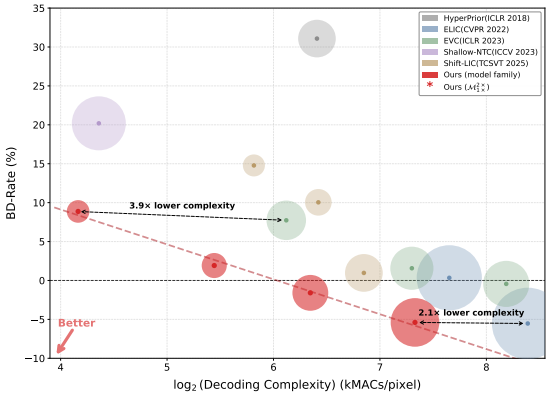


Figure 1: Complexity-performance trade-off on Kodak (BD-rate relative to VTM-24.0 versus decoding kMACs/pixel; bubble size: parameter count). Our model family spans multiple budgets and improves the trade-off over existing methods.

Existing low-complexity codecs typically reduce costs by employing shallow decoders, simple entropy models, or lightweight operations (Yang & Mandt, 2023; Zhang et al., 2025b; He et al., 2026). While effective, these designs are tailored to a single, fixed complexity level and lack a systematic methodology for constructing models across varying budgets. This raises a critical question: does an architecture that performs well under one budget become suboptimal under another? Can we devise a practical strategy to consistently yield high-performing models across diverse complexity constraints?

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

In this paper, we formulate learned image compression as budget-constrained model design and provide a practical solution. We first design a unified lightweight architecture adaptable to varying complexity budgets by adjusting its internal components. Using edge-device constraints as a baseline, we investigate how different complexity allocations between the entropy model and the transform affect rate-distortion performance, thereby determining a coarse-grained complexity distribution. Building on these findings, we conduct local structure search and composition for fine-grained tuning. Starting from this optimized structure, we expand it into a model family via hierarchical pruning from the largest-capacity model, further boosting performance. Through this pipeline, we not only obtain a high-performing model for a given budget but also derive a comprehensive model family spanning diverse complexity levels. As shown in Figure 1, our model family outperforms existing methods across different complexity budgets. In summary, our contributions are:

- **Unified lightweight architecture.** We propose a unified lightweight architecture adaptable to diverse complexity budgets by flexibly adjusting its internal components.
- **Budget-guided complexity allocation.** Coarse-grained allocation between the entropy model and transform, followed by fine-grained local structure search and composition, yields a strong structure at low training cost.
- **Hierarchical pruning for further gains.** We incorporate a hierarchical pruning strategy to further enhance model performance.
- **Consistent superiority across budgets.** Extensive experiments demonstrate that our model family consistently outperforms existing methods across complexity budgets.

2 Related Work

Learned Image Compression. End-to-end learned compression builds on nonlinear transform coding (Ballé et al., 2017; 2021). Entropy models have progressed through scale hyperpriors (Ballé et al., 2018), joint autoregressive–hierarchical priors (Minnen et al., 2018), Gaussian mixtures with attention (Cheng et al., 2020), and generalized Gaussian likelihoods (Zhang et al., 2025a). Transform and context designs span window-based attention (Zou et al., 2022), mixed CNN–Transformer blocks (Liu et al., 2023), channel-wise and checkerboard conditioning (Minnen & Singh, 2020; He et al., 2021), and uneven or multi-reference grouping (He et al., 2022; Jiang et al., 2023; 2025; Li et al., 2025). Gain and modulation units enable variable-rate operation (Cui et al., 2021; Li et al., 2023). These works define the transform-and-entropy framework within which we study capacity allocation.

Low-Complexity Neural Codecs. Johnston et al. (2019) study automatic network optimization and decoder runtime. Shallow decoders shift capacity toward the encoder (Yang & Mandt, 2023), while lightweight codecs combine efficient convolutions with simplified probability models (Zhang et al., 2025b; He et al., 2026). On-device deployment and standardization efforts further reflect practical complexity constraints (Tatwawadi et al., 2026; Ascenso et al., 2023). Our focus is a bitrate-dependent compute-allocation principle under tight decoding budgets and its use in joint transform–entropy design, rather than efficient architecture design alone.

3 Methodology

3.1 Overview

We consider learned image compression under a prescribed computational budget. Let $\mathcal{A}$ denote the architecture space induced by a unified set of construction rules, $\boldsymbol{a} \in \mathcal{A}$ an architecture, $\boldsymbol{w}$ its learnable parameters, and $\boldsymbol{s}$ a training strategy. For a training set $\mathbb{X}_{\mathrm{tr}}$ and a validation set $\mathbb{X}_{\mathrm{val}}$, the optimized parameters are

$$\boldsymbol{w}^*(\boldsymbol{a}, \boldsymbol{s}) = \mathrm{Train}\left(\boldsymbol{a}, \boldsymbol{s}; \mathbb{X}_{\mathrm{tr}}\right). \tag{1}$$

Our objective is to identify an architecture and an optimization strategy that minimize the rate-distortion loss under the complexity constraint:

$$\begin{aligned} \min_{\boldsymbol{a}\in\mathcal{A},\, \boldsymbol{s}\in\mathcal{S}} \quad & \mathcal{L}_{\mathrm{RD}}\left(\boldsymbol{w}^*(\boldsymbol{a}, \boldsymbol{s}), \boldsymbol{a}; \mathbb{X}_{\mathrm{val}}\right) \\ \text{s.t.} \quad & C(\boldsymbol{a}) \le C_{\mathrm{budget}}, \end{aligned} \tag{2}$$

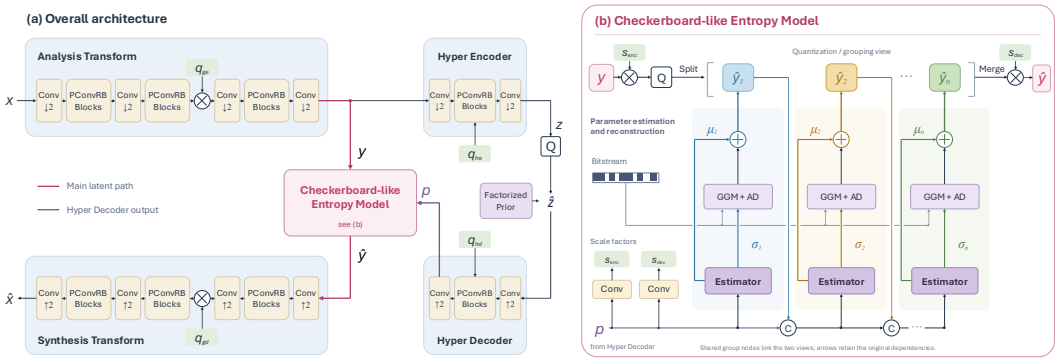


Figure 2: The main architecture of our proposed method. It consists of an analysis transform, a synthesis transform, a hyper prior entropy model, and a proposed checkerboard-like entropy model.

where $C(\boldsymbol{a})$ measures the computational complexity of architecture $\boldsymbol{a}$, and C_{budget} is the target complexity budget. The rate-distortion objective is

$$\mathcal{L}_{\text{RD}} = \mathcal{D}(\boldsymbol{x}, \hat{\boldsymbol{x}}) + \lambda \left(\mathcal{R}(\boldsymbol{y}) + \mathcal{R}(\boldsymbol{z})\right), \tag{3}$$

where $\mathcal{D}(\cdot)$ denotes the distortion. $\mathcal{R}(\cdot)$ denotes the bitrate. $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ are the input and reconstructed images, respectively; $\boldsymbol{y}$ and $\boldsymbol{z}$ denote the main and hyper latents; $R_{\boldsymbol{y}}$ and $R_{\boldsymbol{z}}$ are their estimated rates; and λ controls the rate-distortion trade-off.

Equation 2 gives rise to three central questions:

- How can we construct an architecture space $\mathcal{A}$ that supports different complexity budgets?
- How can we efficiently identify a strong architecture $\boldsymbol{a} \in \mathcal{A}$ under a strict budget?
- Given a selected architecture, how can we design an optimization strategy $\boldsymbol{s}$ that further improves its rate-distortion performance?

We address these questions in three steps. First, we integrate lightweight components from learned image compression into a unified architecture. Second, we analyze computation allocation across major components and perform low-cost local structure search under the target decoding budget. Third, we construct a model family from the resulting compact architecture through gradual expansion and energy-based hierarchical pruning. All models share the same coding framework while covering a broad range of computational budgets.

3.2 Basic Model Construction

As illustrated in Figure 2, the overall architecture supports complexity adjustment along three dimensions: the number of blocks per module, the channel width of intermediate representations, and the number of entropy-model iterations. The design is built upon the following components.

Lightweight Module Design. Under a strict computational budget, the backbone must balance representational capacity with low arithmetic and memory overhead. Inspired by recent lightweight learned compression models (Zhang et al., 2025b; He et al., 2026; Li et al., 2026), we construct the analysis and synthesis transforms using lightweight operations. Details are described in Appendix A.1.

Checkerboard-like Entropy Model. The entropy model is governed by two factors: the number of sequential decoding steps and the subset of latent elements processed per step. More steps provide richer causal context at the cost of higher decoding complexity, while for a fixed step count, performance depends on the partition and ordering of latent elements. We introduce a unified spatial–channel partition that supports multiple configurations. Starting from a hyperprior, we divide the main latent into 16 groups following a spatial-channel interleaved, checkerboard-like pattern. By merging these groups under different schedules, the same formulation supports 1, 2, 4, 8, or 16 sequential decoding steps. This formulation unifies several widely used entropy models (Ballé et al.,

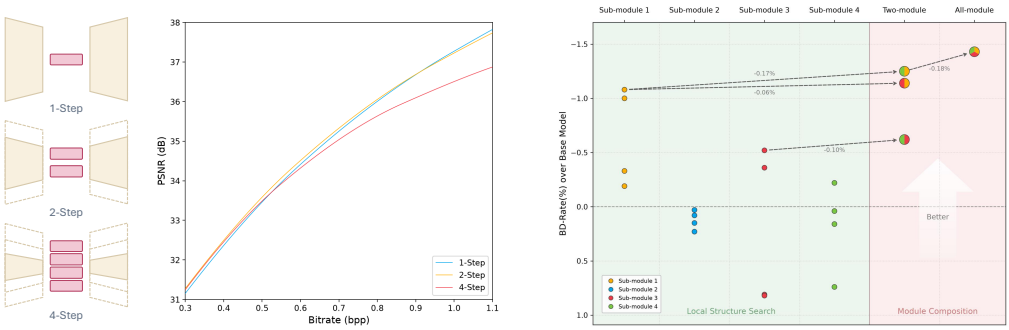


Figure 3: **Left:** Rate-distortion performance curves of 1-, 2-, and 4-Step checkerboard entropy models at a total decoding complexity of nearly 20 kMACs/pixel. **Right:** Performance comparison between the local structure search and module composition model and the baseline model.

2018; He et al., 2021; Li et al., 2023) and provides a configurable search space for balancing compression performance against decoding complexity. For probability estimation, we adopt a generalized Gaussian distribution (Zhang et al., 2025a), which offers a more flexible representation of latent distributions than a single Gaussian while avoiding the overhead of a Gaussian mixture model. Details are described in Appendix A.2.

Variable-Rate Coding. A practical codec should support multiple rate points with a single parameter set, reducing storage overhead and enabling continuous quality adjustment. Following prior variable-rate methods (Cui et al., 2021; Li et al., 2023), we introduce a quality index q to control the operating point. Specifically, q is mapped by four learnable embedding functions and interpolated to produce channel-wise modulation coefficients $q_{ge}, q_{gd}, q_{he}, q_{hd}$ for the analysis, synthesis, hyper-analysis, and hyper-synthesis transforms, respectively. These coefficients modulate the corresponding intermediate representations, allowing one model to adapt its transforms and latent distributions to different quality levels. Details are described in Appendix A.3.

3.3 Architecture Analysis under Budget Constraints

With the unified framework established, we determine a concrete architecture under a target deployment budget. Targeting on-device image coding on mainstream smartphones, we constrain the decoding complexity to approximately 20 kMACs/pixel—roughly half that of PICO (Tatwawadi et al., 2026) and comparable to the range considered by JPEG-AI BOP (Ascenso et al., 2023). Since exhaustive neural architecture search over all budget-compliant configurations is prohibitively expensive, we decompose the problem into two stages: first allocating computation between the transforms and the entropy model, then refining individual modules through a local structure search.

Entropy-Transform Trade-off. The transforms and the entropy model are the two dominant computational components of a learned image codec. Under a tight budget, increasing the capacity of one necessarily reduces the resources available to the other. We therefore study how computation should be allocated between them.

Starting from a 1-step baseline that satisfies the 20 kMACs/pixel budget, we increase the number of entropy-model steps while keeping the cost per step approximately fixed, and proportionally reduce the channel width of the analysis and synthesis transforms to maintain similar overall complexity. This yields 2-step and 4-step variants; we exclude 8-step and 16-step configurations because they either leave insufficient computation for the transforms or exceed the budget.

Figure 3(left) compares the rate-distortion performance of these variants. The 1-step model performs best at high bitrates, but is gradually surpassed by the 2-step and 4-step models as the bitrate decreases, with the 2-step model achieving the strongest performance in the low-to-medium bitrate range. Although the 4-step model underperforms at high bitrates, it surpasses the 1-step model at low bitrates and approaches the 2-step model. These results reveal a bitrate-dependent trade-off: at high bitrates, expressive transforms are more beneficial than stronger context modeling, whereas at low bitrates, improved probability estimation yields larger gains than additional transform capac-

ity. Balancing performance and sequential decoding cost, we adopt the 2-step configuration for the target bitrate range.

Local Structure Search and Module Composition. After determining the high-level allocation between the transform and entropy models, we refine each module on top of the base model. Since a joint search over all module configurations remains costly, we follow the idea in Chen et al. (2025) by decomposing the model into several sub-modules: for each sub-module, we conduct a local search while keeping the others fixed, and after all sub-modules have been searched, we compose the best-performing structure of each into the final model.

We decompose the network into m sub-modules. Let $\mathcal{A}_i$ denote the candidate set for the i-th sub-module with $n_i = |\mathcal{A}_i|$. A joint search requires evaluating $O\left(\prod_{i=1}^{m} n_i\right)$ architectures, whereas optimizing one sub-module at a time reduces this to $O\left(\sum_{i=1}^{m} n_i\right)$. To avoid degenerate computation allocation, each candidate set is constrained to a narrow complexity range around its counterpart in the base architecture. For each sub-module, we identify the locally optimal candidate under a shared short-training protocol. Let $\boldsymbol{a}^{(S)}$ denote the architecture obtained by replacing the sub-modules indexed by $S \subseteq \{1, \ldots, m\}$ with their locally optimal candidates while retaining the base configurations for the remaining modules. Our module-composition hypothesis is

$$\mathcal{L}^{*}_{\mathrm{RD}}\left(\boldsymbol{a}^{(\{1,\ldots,m\})}\right) \leq \mathcal{L}^{*}_{\mathrm{RD}}\left(\boldsymbol{a}^{(S)}\right), \qquad \forall S \subseteq \{1, \ldots, m\}, \tag{4}$$

where $\mathcal{L}^{*}_{\mathrm{RD}}$ denotes the rate–distortion loss after applying the same optimization strategy. In practice, Equation 4 may occasionally be violated when combining locally optimal sub-modules; nevertheless, the proposed strategy reliably identifies an architecture that outperforms the initial design.

We adopt the 2-step model as the base architecture, divide it into four sub-modules, and search them with shortened training schedules. The best candidate per sub-module is then combined and evaluated for compatibility. As shown in Figure 3(right), models with more locally optimized sub-modules perform no worse than their partial combinations. One limitation is that each sub-module inherits its complexity range from the base architecture, which may preclude global optimality; we mitigate this by constructing the base model following established low-complexity designs (Zhang et al., 2025b; He et al., 2026), yielding a strong initial computation allocation. The resulting architecture is a practical, high-performing solution under the target budget, though not guaranteed to be globally optimal. More details are described in Appendix A.5

3.4 Model Family Construction via Expansion and Hierarchical Pruning

The preceding analysis yields a compact model, denoted by $\mathcal{M}_{1\times}$. We next construct a scalable model family and improve its compact members by transferring knowledge from higher-capacity models. Such benefits have been observed in learned image compression through pruning and distillation (Wang et al., 2023; Chen et al., 2025).

Gradual Model Expansion. We begin with $\mathcal{M}_{1\times}$, the compact architecture from the preceding analysis. Rather than designing larger models independently, we progressively increase its capacity to construct $\mathcal{M}_{2\times}$, $\mathcal{M}_{4\times}$, and $\mathcal{M}_{8\times}$. All members share the same coding structure and differ primarily in depth and channel width. Each expanded architecture records an explicit correspondence between its blocks and those of the preceding model, providing the structural mapping required for reverse parameter transfer.

Gradual expansion serves two purposes. First, it constructs a scalable family without redesigning the coding framework for each complexity level. Second, it ensures structural compatibility between adjacent models, making parameter inheritance more reliable than mapping between independently designed architectures.

Energy-based Hierarchical Pruning. Starting from the trained $\mathcal{M}_{8\times}$, we construct compact models along the reverse hierarchy

$$\mathcal{M}_{8\times} \xrightarrow{\text{prune+finetune}} \mathcal{M}_{4\times} \xrightarrow{\text{prune+finetune}} \mathcal{M}_{2\times} \xrightarrow{\text{prune+finetune}} \mathcal{M}_{1\times}. \tag{5}$$

At each stage, the current model initializes the adjacent smaller architecture. For layers with mismatched channel dimensions, we rank channels by activation energy and retain the top-K to initial-

ize the student. For depth mismatches, we selectively inherit a subset of teacher modules. Details are provided in Appendix B.2.

4 Experimental Results

4.1 Experiment Settings

Training Settings. We combine the training sets of CLIC-2020 [1], DIV2K (Agustsson & Timofte, 2017), and Flickr2K (Timofte et al., 2017) to form our training set. During training, images are cropped to 512×512 patches with a batch size of 8. The models are optimized by Equation 3. We use mean square error (MSE) as the distortion metric. The Lagrange multiplier λ takes four possible values $\{2.25, 1.16, 0.67, 0.38\}$ during training, where each image in a training batch is randomly assigned one of these values as its λ. The Adam (Kingma & Ba, 2014) optimizer is used with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Our models are optimized with 2 million training steps. The learning rate is set as 1×10^{-4} initially, and reduced to 2×10^{-5} after 1.6 million steps, then to 5×10^{-6} after 1.8 million steps, and then to 1×10^{-6} after 1.9 million steps.

Model Settings. We adopt $\mathcal{M}_{1\times}$ as the base architecture and derive $\mathcal{M}_{2\times}$, $\mathcal{M}_{4\times}$, and $\mathcal{M}_{8\times}$ for hierarchical pruning. We further construct $\mathcal{M}_{1\times}^{2\times}$ by combining the encoder of $\mathcal{M}_{2\times}$ with the decoder of $\mathcal{M}_{1\times}$, to isolate the gains from increasing encoding complexity under a fixed decoding budget. To examine scalability, we design $\mathcal{M}_{36\times}$ with a 16-step entropy model, and to validate performance under tighter constraints, we develop $\mathcal{M}_{0.5\times}$ and $\mathcal{M}_{0.25\times}$. Together, these models span a wide range of complexities; detailed specifications are provided in Appendix A.4.

Evaluation Settings. We evaluate on two datasets: Kodak [2] (24 images at 512×768) and CLIC-PV [3] (CLIC 2020 professional validation, 41 high-quality images). Bitrate is measured in bits-per-pixel (bpp) and distortion in PSNR; we also report BD-rate (Bjontegaard, 2001) with VTM-24.0 [4] as the anchor. Coding time is measured on an NVIDIA L20 GPU. Model complexity (kMACs/pixel and parameter count) is evaluated using DeepSpeed [5]. We compare against: (1) model-family methods supporting multiple complexity scales—ELIC (He et al., 2022), EVC (Wang et al., 2023), JPEG-AI (Ascenso et al., 2023), ShiftLIC (Bao et al., 2025); (2) low-decoding-complexity models—HyperPrior (Ballé et al., 2018), Shallow-NTC (Yang & Mandt, 2023), FastNIC (Zhang et al., 2025b), He2026 (He et al., 2026); and (3) high-complexity models—MLIC++ (Jiang et al., 2025), FLIC (Li et al., 2024), LALIC (Feng et al., 2025), HPCM (Li et al., 2025).

4.2 Rate-distortion Performance

Performance Under Primary Complexity. At a decoder complexity budget of 20 kMACs/pixel, we evaluate two primary models, $\mathcal{M}_{1\times}$ and $\mathcal{M}_{1\times}^{2\times}$. As shown in Table 5, they incur BD-rate losses of 8.87% / 9.18% and 7.96% / 7.91% against VTM on the Kodak and CLIC-PV datasets, respectively. Both models outperform existing low-complexity approaches with lower encoding and decoding costs. Notably, compared to Shallow-NTC (Yang & Mandt, 2023), which has comparable decoder complexity, our $\mathcal{M}_{1\times}^{2\times}$ achieves a 24.82% BD-rate gain while reducing encoding and decoding complexity by 84% and 13%, respectively. To the best of our knowledge, this establishes a new state-of-the-art at the 20 kMACs/pixel level.

Performance Across Model Families of Varying Complexity. Through gradual model expansion and hierarchical pruning, we obtain a scalable model family ($\mathcal{M}_{1\times}$, $\mathcal{M}_{2\times}$, $\mathcal{M}_{4\times}$, $\mathcal{M}_{8\times}$) sharing an identical backbone while spanning multiple complexity levels. As shown in Table 5 and Figure 4, it achieves state-of-the-art rate-distortion performance at each operating point, outperforming existing alternatives. Analogous to EVC (Wang et al., 2023), our family natively supports variable bitrate for flexible deployment. Notably, Figure 1 reveals an empirical regularity: *with a fixed backbone,*

[1] http://compression.cc
[2] https://r0k.us/graphics/kodak/
[3] http://compression.cc
[4] https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM
[5] https://github.com/deepspeedai/DeepSpeed

Table 1: Rate-distortion performance and complexity comparison. The anchor is VTM-24.0. BD-rate is computed in terms of PSNR, where lower values indicate better performance. VBR denotes variable bitrate support. Time is computed on 1920×1088 images.

Method	VBR	Encoding		Decoding		Param.	BD-rate(%)*	
		complexity (kMACs/pixel)	time (ms)	complexity (kMACs/pixel)	time (ms)	(M)	Kodak	CLIC-PV
HM-18.0	✓	-	-	-	-	-	+25.31	+30.95
HyperPrior	✗	88.0	118.8	84.9	126.2	11.58	+31.08	+39.55
Shallow-NTC	✗	277.3	69.8	20.5	6.3	23.85	+32.78	+48.43
JPEG-AI BOP	✓	48.5	26.1	23.1	60.1	12.20	+58.97	+52.47
JPEG-AI HOP	✓	255.6	193.5	214.2	216.5	16.55	+39.12	33.56
ELIC-SM	✗	204.3	-	201.1	-	34.77	+0.34	-
ELIC	✗	338.4	95.4	335.1	91.3	41.94	-5.53	-4.88
EVC-S	✓	71.1	47.3	69.6	35.6	12.78	+7.73	+9.58
EVC-M	✓	161.5	90.1	157.6	56.0	15.23	+1.57	+2.78
EVC-L	✓	298.5	149.5	291.3	85.2	17.38	-0.44	+0.49
Shift-LIC-Small	✗	57.7	42.0	56.3	41.5	3.97	+14.78	+13.32
Shift-LIC-Middle	✗	87.1	54.5	85.7	54.2	5.79	+10.04	+6.85
Shift-LIC-Large	✗	116.7	140.9	115.3	203.6	11.65	+0.97	-4.80
$\mathcal{M}_{1\times}$(Ours)	✓	19.1	9.9	17.9	9.7	4.19	+8.87	+9.18
$\mathcal{M}_{2\times}$(Ours)	✓	44.6	25.5	43.5	25.6	5.12	+1.92	+0.85
$\mathcal{M}_{4\times}$(Ours)	✓	83.5	35.9	81.4	36.1	10.51	-1.59	-3.17
$\mathcal{M}_{8\times}$(Ours)	✓	162.9	57.3	160.8	57.8	19.32	-5.39	-7.39
$\mathcal{M}_{1\times}^{2\times}$(Ours)	✓	44.6	25.5	17.9	9.8	4.67	+7.96	+7.91

* BD-rate are computed within the bitrate range of our method.

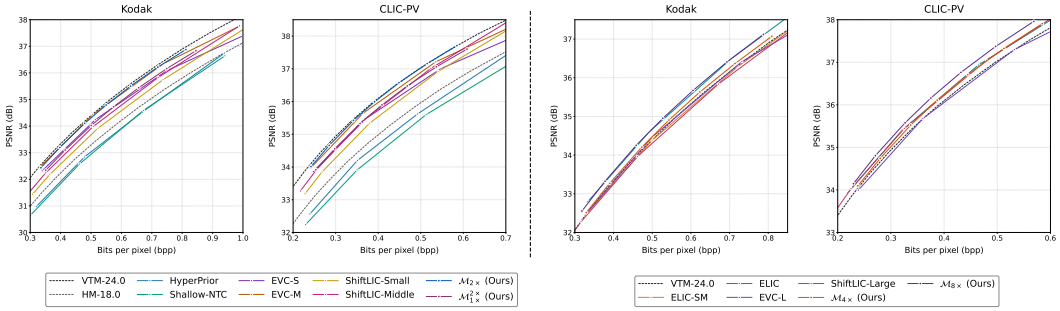


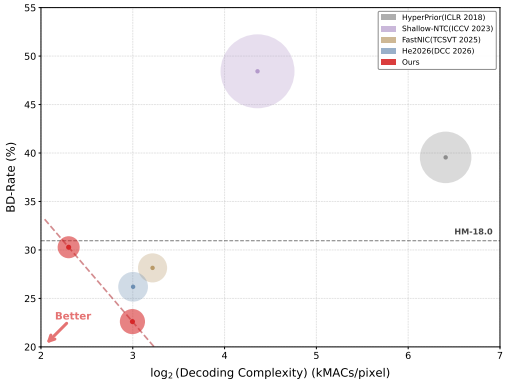
Figure 4: Rate-distortion performance on the Kodak and CLIC-PV datasets. The left two subfigures compare the proposed $\mathcal{M}_{2\times}$ and $\mathcal{M}_{1\times}^{2\times}$ configurations with representative conventional and learned image codecs. The right two subfigures compare the larger $\mathcal{M}_{4\times}$ and $\mathcal{M}_{8\times}$ configurations with stronger learned compression baselines.

performance scales approximately linearly while complexity grows exponentially across the family. We note that further validation across diverse architectures and tasks is needed to establish its generalizability.

Performance Under Tighter Complexity Budgets. To further investigate our approach under tighter complexity budgets, we construct $\mathcal{M}_{0.5\times}$ with decoding complexity constrained to 8 kMACs/pixel. Entropy-transform trade-off analysis shows that a 2-step entropy model is infeasible at this level, so we simplify to a 1-step design. This yields the observation that *the optimal architecture varies with complexity budget and bitrate*—a point rarely systematically explored despite its intuitiveness. Building on $\mathcal{M}_{0.5\times}$, we further derive $\mathcal{M}_{0.25\times}$ by reducing capacity to meet a 5 kMACs/pixel budget. Results in Figure 5(left) and Table 2 show that, compared to He2026 (He et al., 2026) at a similar complexity regime, our $\mathcal{M}_{0.5\times}$ achieves gains of 0.39% and 3.59% on

Table 2: Rate-distortion performance and complexity comparison under tighter budgets. The anchor is VTM-24.0. Time is computed on 1920×1088 images. Since the source code for FastNIC and He2026 is not publicly available, we do not measure their times.

Method	Encoding		Decoding		Param.	BD-rate(%)	
	complexity (kMACs/pixel)	time (ms)	complexity (kMACs/pixel)	time (ms)	(M)	Kodak	CLIC-PV
HM-18.0	-	-	-	-	-	+25.31	+30.95
JPEG-AI SOP	48.50	25.9	7.91	54.9	12.20	+78.40	+67.67
FastNIC	11.02	-	9.29	-	3.70	+21.97	+28.15
He2026	8.37	-	8.02	-	3.86	+21.43	+26.20
$\mathcal{M}_{0.5\times}$(Ours)	9.08	5.6	7.98	5.3	2.76	**+21.04**	**+22.61**
$\mathcal{M}_{0.25\times}$(Ours)	5.85	3.8	4.93	3.5	2.12	+26.62	+30.28



Ablation Study	BD-rate (%)	Δ
$\mathcal{M}_{1\times}$ w/o VBR&pruning	+10.04	-
+ variable bitrate	+11.03	+0.99
+ single-step pruning	+9.10	-0.94
+ equal training epoch	+9.94	-0.10
+ $2\times$ training epoch	+8.94	-1.10
+ full pruning ($\mathcal{M}_{1\times}$)	**+8.87**	**-1.17**
+ large encoder ($\mathcal{M}_{1\times}^{2\times}$)	**+7.96**	**-2.08**

Figure 5: Low-complexity performance and ablation study. **Left:** Complexity-performance trade-off on CLIC-PV. The horizontal dashed line indicates HM-18.0; lower-left is better. **Right:** Ablation study of the proposed $\mathcal{M}_{1\times}$ model on Kodak.

Kodak and CLIC-PV, respectively, with more pronounced advantages on high-resolution images. Meanwhile, $\mathcal{M}_{0.25\times}$ performs comparably to HM-18.0 [6] on CLIC-PV.

High-Complexity Model Performance. To explore the upper bound of our architecture, we construct $\mathcal{M}_{36\times}$ with a total forward-pass complexity of approximately 1000 kMACs/pixel. This model scales capacity by widening channels and deepening layers across all modules, and employs a 16-step entropy model. As shown in Table 3, our convolution-friendly design enables $\mathcal{M}_{36\times}$ to achieve the shortest inference latency among models at a similar complexity level. In terms of rate-distortion performance, it surpasses MLIC++ and FLIC only on the CLIC-PV dataset. We attribute this limitation to two factors: (1) adopting a unified architecture across complexity levels precludes integrating computationally intensive modules such as Transformers; and (2) the high training cost makes it difficult to conduct architecture analysis and hierarchical pruning for this scenario, so the reported performance reflects that of an initialized model.

4.3 Visual Comparison

Figure 6 presents a visual comparison of reconstructed images produced by our primary-complexity model against those of competing methods. Our approach achieves higher PSNR at a lower bitrate, yielding visibly superior reconstruction quality.

4.4 Ablation Study

Impact of Variable-Rate. To assess the impact of variable bitrate on performance, we conduct an additional experiment using fixed-rate models. Specifically, rather than randomly sampling within

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

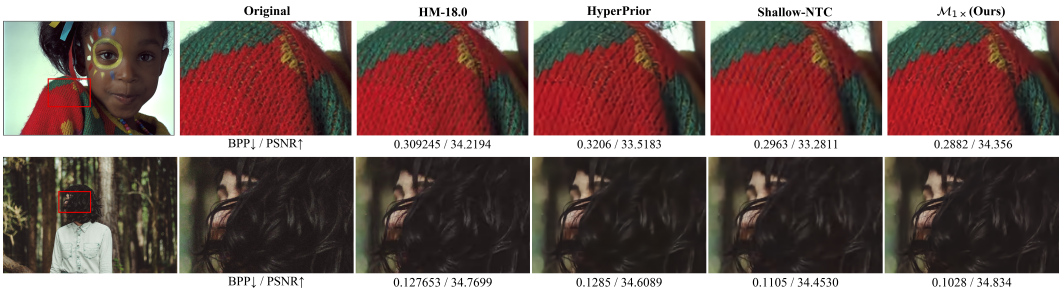


Figure 6: Visual comparison for our $\mathcal{M}_{1\times}$ model. Compared with existing methods, our approach achieves lower BPP and higher PSNR with reduced computational complexity.

Table 3: Rate-distortion performance and complexity comparison for high-complexity settings. The anchor is VTM-24.0. Time is computed on 768×512 images. Complexity and time are both computed using the total forward process.

Method	**Complexity** (kMACs/pixel)	**Time** (ms)	**Param.** (M)	**BD-rate(%)** Kodak	CLIC-PV
MLIC++	1282.8	62.6	116.7	-11.74	-11.02
FLIC	1096.0	79.7	71.0	-12.67	-11.63
LALIC	>675 †	105.4	66.1	-13.46	-15.84
HPCM	918.6	33.8	68.5	-13.86	-14.35
$\mathcal{M}_{36\times}$(Ours)*	966.3	29.0	144.4	-10.49	-12.70

† The MACs of some operators cannot be computed using the DeepSpeed library.
* Baseline performance **w/o architecture analysis and hierarchical pruning**.

each image in a training batch, we fix λ to a constant value throughout training. The results are presented in Figure 5(right). Compared to the fixed bitrate setting, employing variable bitrate incurs a 0.99% performance degradation.

Effectiveness of Hierarchical Pruning. To validate hierarchical pruning, we conduct three controlled comparisons: (1) single-step pruning from $\mathcal{M}_{8\times}$ to $\mathcal{M}_{1\times}$ with extended training to match the total computational budget; (2) from-scratch training of $\mathcal{M}_{1\times}$ extended to match the total training cost; and (3) from-scratch training extended to twice the iterations of our hierarchical pruning. As shown in Figure 5(right), while both single-step pruning and extended training yield non-trivial improvements, hierarchical pruning consistently achieves the best performance. Notably, from-scratch training at twice the iterations attains performance comparable to our method, suggesting that pruning primarily accelerates convergence. Beyond this, hierarchical pruning naturally yields a model family spanning multiple complexity levels—a structural advantage unavailable to single-step pruning or extended training.

5 Conclusion

We present a systematic study of learned image compression model design under prescribed complexity budgets. We introduce a practical budget-constrained design procedure built on a unified lightweight architecture with entropy-transform trade-off, local structure search, and module composition. Starting from a compact base model, we progressively expand its capacity and then apply energy-based hierarchical pruning to construct a structurally compatible model family. Extensive experiments demonstrate that our model family consistently outperforms existing methods across decoding budgets between 5-160 kMACs/pixel. Although specialized high-complexity architectures remain advantageous in that regime, our results establish an effective and generalizable approach to budget-aware learned compression design. Future work will extend this framework to broader budget ranges and incorporate additional practical constraints, such as latency, memory footprint, and hardware-specific deployment considerations.

AI USE STATEMENT

We used AI to assist with language editing, drafting and revising portions of the Introduction, Related Work, and Conclusion, literature discovery, reference formatting, and LaTeX preparation. We also used it to obtain feedback on the research framing, the interpretation of existing results, and the design of ablation studies and controlled comparisons. These uses did not include generating experimental measurements or synthetic datasets, or proving mathematical claims. The authors are responsible for verifying all AI-assisted material and for the final content of the paper.

REFERENCES

Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *2017 IEEE conference on computer vision and pattern recognition workshops (CVPRW)*, pp. 1122–1131. IEEE, 2017.

João Ascenso, Elena Alshina, and Touradj Ebrahimi. The jpeg ai standard: Providing efficient human and machine visual data consumption. *Ieee Multimedia*, 30(1):100–111, 2023.

Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end optimized image compression. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1611.01704.

Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *International Conference on Learning Representations*, 2018. URL https://arxiv.org/abs/1802.01436.

Jona Ballé, Philip A. Chou, David Minnen, Saurabh Singh, Nick Johnston, Eirikur Agustsson, Sung Jin Hwang, and George Toderici. Nonlinear transform coding. *IEEE Journal of Selected Topics in Signal Processing*, 15(2):339–353, 2021. doi: 10.1109/JSTSP.2020.3034501. URL https://arxiv.org/abs/2007.03034.

Youneng Bao, Wen Tan, Chuanmin Jia, Mu Li, Yongsheng Liang, and Yonghong Tian. Shiftlic: Lightweight learned image compression with spatial-channel shift operations. *IEEE Transactions on Circuits and Systems for Video Technology*, 35(9):9428–9442, 2025.

Gisle Bjontegaard. Calculation of average psnr differences between rd-curves. *ITU SG16 Doc. VCEG-M33*, 2001.

Yunuo Chen, Zezheng Lyu, Bing He, Ning Cao, Gang Chen, Guo Lu, and Wenjun Zhang. Knowledge distillation for learned image compression. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4996–5006. IEEE, 2025.

Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized Gaussian mixture likelihoods and attention modules. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. URL https://arxiv.org/abs/2001.01568.

Ze Cui, Jing Wang, Shangyin Gao, Tiansheng Guo, Yihui Feng, and Bo Bai. Asymmetric gained deep image compression with continuous rate adaptation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10527–10536. IEEE, 2021.

Donghui Feng, Zhengxue Cheng, Shen Wang, Ronghua Wu, Hongwei Hu, Guo Lu, and Li Song. Linear attention modeling for learned image compression. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pp. 7623–7632, June 2025. URL https://arxiv.org/abs/2502.05741.

Andrea Giachetti. Testimages: a large-scale archive for testing visual devices and basic image processing algorithms. 2014.

Dailan He, Yaoyan Zheng, Baocheng Sun, Yan Wang, and Hongwei Qin. Checkerboard context model for efficient learned image compression. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14766–14775. IEEE, 2021.

Dailan He, Ziming Yang, Weikun Peng, Rui Ma, Hongwei Qin, and Yan Wang. 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*, 2022. URL https://arxiv.org/abs/2203.10886.

Qian He, Yuqi Li, Yiheng Jiang, Haotian Zhang, Li Li, and Dong Liu. Practical image compression with energy-guided asymmetric entropy modeling. In *2026 Data Compression Conference (DCC)*, pp. 292–301, 2026. doi: 10.1109/DCC66757.2026.00037.

Wei Jiang, Jiayu Yang, Yongqi Zhai, Peirong Ning, Feng Gao, and Ronggang Wang. MLIC: Multi-reference entropy model for learned image compression. In *Proceedings of the 31st ACM International Conference on Multimedia*, pp. 7618–7627, 2023. doi: 10.1145/3581783.3611694. URL https://arxiv.org/abs/2211.07273.

Wei Jiang, Jiayu Yang, Yongqi Zhai, Feng Gao, and Ronggang Wang. MLIC++: Linear complexity multi-reference entropy modeling for learned image compression. *ACM Transactions on Multimedia Computing, Communications and Applications*, 21(5):142:1–142:25, 2025. doi: 10.1145/3719011. URL https://arxiv.org/abs/2307.15421.

Nick Johnston, Elad Eban, Ariel Gordon, and Johannes Ballé. Computationally efficient neural image compression. *arXiv preprint arXiv:1912.08771*, 2019. URL https://arxiv.org/abs/1912.08771.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Han Li, Shaohui Li, Wenrui Dai, Chenglin Li, Junni Zou, and Hongkai Xiong. Frequency-aware transformer for learned image compression. In *International conference on learning representations*, volume 2024, pp. 30447–30465, 2024.

Jiahao Li, Bin Li, and Yan Lu. Neural video compression with diverse contexts. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22616–22626. IEEE, 2023.

Jiahao Li, Wenxuan Xie, Zhaoyang Jia, Bin Li, Zongyu Guo, Xiaoyi Zhang, and Yan Lu. Ultra-fast neural video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 41311–41321, June 2026.

Yuqi Li, Haotian Zhang, Li Li, and Dong Liu. Learned image compression with hierarchical progressive context modeling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025. URL https://arxiv.org/abs/2507.19125.

Jinming Liu, Heming Sun, and Jiro Katto. Learned image compression with mixed Transformer-CNN architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. URL https://arxiv.org/abs/2303.14978.

David Minnen and Saurabh Singh. Channel-wise autoregressive entropy models for learned image compression. In *IEEE International Conference on Image Processing*, 2020. doi: 10.1109/ICIP40778.2020.9190935. URL https://arxiv.org/abs/2007.08739.

David Minnen, Johannes Ballé, and George Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://arxiv.org/abs/1809.02736.

Kedar Tatwawadi, Parisa Rahimzadeh, Zhanghao Sun, Zhiqi Chen, Ziyun Yang, Sanjay Nair, Divija Hasteer, and Oren Rippel. What matters in practical learned image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12095–12105, June 2026.

Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, Lei Zhang, Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, Kyoung Mu Lee, et al. Ntire 2017 challenge on single image super-resolution: Methods and results. In *2017 IEEE conference on computer vision and pattern recognition workshops (CVPRW)*, pp. 1110–1121. IEEE, 2017.

Guo-Hua Wang, Jiahao Li, Bin Li, and Yan Lu. EVC: Towards real-time neural image compression with mask decay. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XUxad2Gj40n.

Yibo Yang and Stephan Mandt. Computationally-efficient neural image compression with shallow decoders. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 530–540. IEEE, 2023.

Haotian Zhang, Li Li, and Dong Liu. Generalized gaussian model for learned image compression. *IEEE Transactions on Image Processing*, 34:1950–1965, 2025a. doi: 10.1109/TIP.2025.3550013.

Haotian Zhang, Yuqi Li, Li Li, and Dong Liu. Learning switchable priors for neural image compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 35(11):11567–11582, 2025b.

Renjie Zou, Chunfeng Song, and Zhaoxiang Zhang. The devil is in the details: Window-based attention for image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. URL https://arxiv.org/abs/2203.08450.

A Model Details

A.1 Lightweight Modules

Up- and Down-sampling Layers. Consistent with Zhang et al. (2025b) and He et al. (2026), we employ 2×2 convolutions with a stride of 2 as our up- and down-sampling layers to reduce computational complexity. The number of channels in this module serves as a tunable option to accommodate various complexity budgets.

Processing Layers in Transforms. We construct the intermediate processing modules of the transform using partial convolution-based residual blocks (Zhang et al., 2025b; Li et al., 2025), which maintain sufficient representation capability while reducing complexity. The number of channels, the number of layers, and the MLP channel ratio of the partial convolution in this module serve as tunable options to accommodate various complexity budgets.

Processing Layers in Entropy Models. In the entropy model, the estimator modules adopt residual blocks composed of depthwise and 1×1 pointwise convolutions (Li et al., 2023; 2026). These designs substantially reduce computation while preserving spatial and channel-wise feature modeling. To allow finer-grained adjustment of the module's complexity, we incorporate a 1×1 channel down-sampling convolution before the standard depth-wise residual blocks. The channel width, layer depth, down-sampling ratio, and the channel ratio of the processing layers serve as tunable options to accommodate various complexity budgets.

A.2 Checkerboard-like Entropy Model

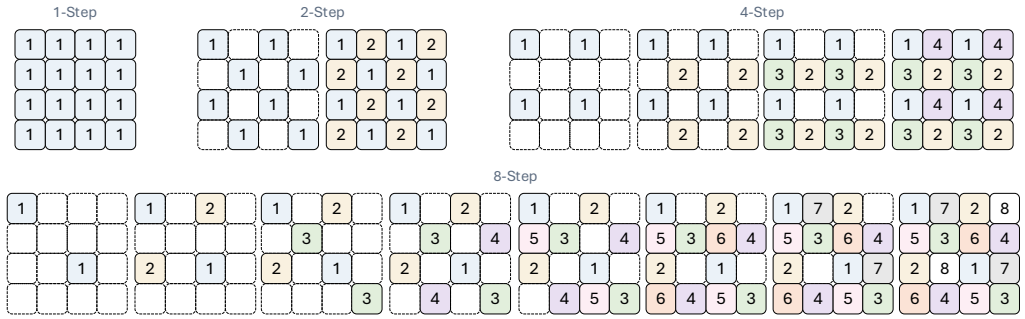


Figure 7: The proposed checkerboard-like entropy model with 1-, 2-, 4-, and 8-step.

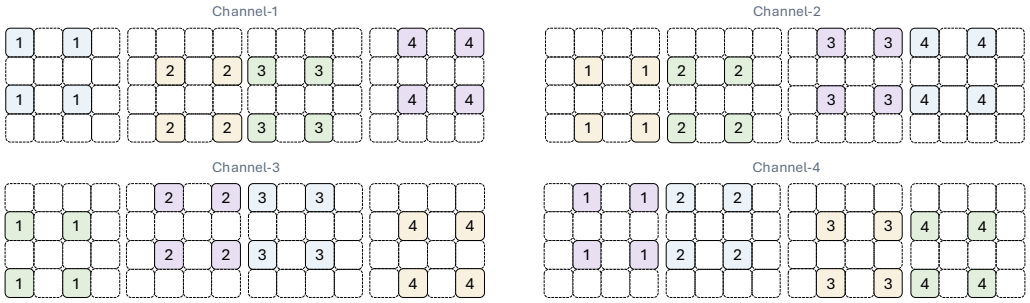


Figure 8: Channel-wise interleaved scheme for 4-step model.

Figure 10 illustrates the element splitting scheme of our checkerboard-like entropy model for 1-, 2-, 4-, and 8-step. The 16-step scenario is not shown but can be directly deduced from the underlying

pattern. Since each level splits every step of the previous level into two in a checkerboard fashion, our construction theoretically supports checkerboard-like entropy models with an arbitrary number of steps that are integer powers of two.

Regarding the iterative order at the channel level, we adopt an interleaved scheme where the first element iterated in the previous channel becomes the last in the next channel, with the remaining elements shifting forward by one position. This strategy maximizes the spatial-channel coverage in each iteration. Figure 8 shows the channel-wise interleaved scheme for 4-step entropy model.

Specifically, the 1-step configuration processes all latent elements in parallel conditioned only on the hyperprior, reducing to the hyperprior-based entropy model (Ballé et al., 2018); the 2-step configuration recovers the standard checkerboard context model (He et al., 2021); and the 4-step configuration follows the spatial–channel pattern of DCVC-DC (Li et al., 2023).

A.3 Variable-Rate Coding

We enable multiple rate-distortion operating points using four sets of learnable channel-wise gain vectors, without changing the main transform and entropy-model parameters. Given L predefined rate levels, the model learns

$$\mathbf{G}^{ge} \in \mathbb{R}^{L\times C_a}, \quad \mathbf{G}^{he} \in \mathbb{R}^{L\times N}, \quad \mathbf{G}^{hd} \in \mathbb{R}^{L\times N}, \quad \mathbf{G}^{gd} \in \mathbb{R}^{L\times C_s}, \tag{6}$$

All gains are initialized to one and are broadcast over spatial dimensions for channel-wise feature modulation.

For a rate index $q \in [0, L-1]$, which can be fractional, we interpolate two adjacent gain vectors in the logarithmic domain. Let

$$l = \lfloor q \rfloor, \qquad u = \min(l+1, L-1), \qquad \alpha = q - l. \tag{7}$$

For a gain bank $\mathbf{G}$, the interpolated gain is

$$\mathcal{I}(\mathbf{G}, q) = |\mathbf{G}_{l,:}|^{1-\alpha} \odot |\mathbf{G}_{u,:}|^{\alpha}. \tag{8}$$

Equivalently, Equation 8 performs linear interpolation on the logarithm of gain magnitudes. This geometric interpolation provides smooth continuous rate control between trained discrete operating points. We define four channel-wise modulation coefficients:

$$q_{ge} = \mathcal{I}(\mathbf{G}^{ge}, q), \quad q_{he} = \mathcal{I}(\mathbf{G}^{he}, q), \quad q_{hd} = \mathcal{I}(\mathbf{G}^{hd}, q), \quad q_{gd} = \mathcal{I}(\mathbf{G}^{gd}, q) \tag{9}$$

The interpolated gains are inserted into the intermediate features of the transforms:

$$\mathbf{y} = \mathcal{G}_{a,2}\left(\mathcal{G}_{a,1}(\mathbf{x}) \odot q_{ge}\right), \tag{10}$$
$$\mathbf{z} = \mathcal{H}_{a,2}\left(\mathcal{H}_{a,1}(\mathbf{y}) \odot q_{he}\right), \tag{11}$$
$$\boldsymbol{\psi} = \mathcal{H}_{s,2}\left(\mathcal{H}_{s,1}(\hat{\mathbf{z}}) \odot q_{hd}\right), \tag{12}$$
$$\hat{\mathbf{x}} = \mathcal{G}_{s,2}\left(\mathcal{G}_{s,1}(\hat{\mathbf{y}}_{\mathrm{dec}}) \odot q_{gd}\right). \tag{13}$$

Here, $\mathcal{G}_a$, $\mathcal{G}_s$, $\mathcal{H}_a$, and $\mathcal{H}_s$ denote the image analysis, image synthesis, hyper-analysis, and hyper-synthesis transforms, respectively.

In addition to the global rate-dependent gains, the hyper-synthesis output $\boldsymbol{\psi}$ predicts content-adaptive encoder- and decoder-side quantization steps:

$$S_{enc} = \max(f_{\mathrm{enc}}(\boldsymbol{\psi}), 0.1), \qquad S_{dec} = \max(f_{\mathrm{dec}}(\boldsymbol{\psi}), 0.1). \tag{14}$$

The main latent is scaled before quantization and restored after decoding:

$$\mathbf{y}_{enc} = \mathbf{y} \odot S_{enc}, \qquad \hat{\mathbf{y}}_{\mathrm{dec}} = \hat{\mathbf{y}}_{enc} \odot S_{dec}. \tag{15}$$

Therefore, the gain vectors control the global target bitrate, while the predicted quantization steps further adapt the coding precision to local image content.

A.4 Detailed Configuration for the Model Family

- M: channel number for the main latent.
- N: channel number for the hyperprior latent.
- G_A: analysis transform.
- G_S: synthesis transform.
- H_A: hyper encoder.
- H_S: hyper decoder.
- EM: checkerboard-like entropy model.
- HC: channel numbers for the hidden layers.
- RN: layer numbers for hidden layers.
- MR: multilayer perceptron ratio for partial convolution or depth-wise convolution based residual blocks.
- PR: partial ratio for partial convolution based residual blocks in transforms.
- STEP: step setting for the checkerboard-like entropy model.
- DR: down-sample ratio for depth-wise convolution based residual blocks.
- LN: layer numbers for each step in entropy models.

Table 4: Detailed configurations of our model family.

Model	M	N	G_A	G_S	H_A	H_S	EM
$M_{0.25\times}$	248	168	HC: [16,32,96] RN: [1,1,1] MR: [4,4,4] PR: [4,4,4]	HC: [16,32,96] RN: [1,1,1] MR: [4,4,4] PR: [4,4,4]	RN: [1,1] MR: [4,4] PR: [4,4]	RN: [1,1] MR: [4,4] PR: [4,4]	STEP: 1 MR: [2] DR: [2] LN: [1]
$M_{0.5\times}$	256	192	HC: [24,48,128] RN: [1,1,1] MR: [4,4,4] PR: [4,4,4]	HC: [24,48,128] RN: [1,1,1] MR: [4,4,4] PR: [4,4,4]	RN: [1,1] MR: [4,4] PR: [4,4]	RN: [1,1] MR: [4,4] PR: [4,4]	STEP: 1 MR: [2] DR: [2] LN: [1]
$M_{1\times}$	256	192	HC: [24,64,128] RN: [2,1,3] MR: [4,4,4] PR: [4,4,4]	HC: [24,64,128] RN: [2,1,3] MR: [4,4,4] PR: [4,4,4]	RN: [1,1] MR: [4,4] PR: [4,4]	RN: [1,1] MR: [4,4] PR: [4,4]	STEP: 2 MR: [1.3*2] DR: [2.7*2] LN: [2*2]
$M_{1\times}^{2\times}$	256	192	HC: [48,80,128] RN: [3,3,5] MR: [4,4,4] PR: [4,4,4]	HC: [24,64,128] RN: [2,1,3] MR: [4,4,4] PR: [4,4,4]	RN: [1,1] MR: [4,4] PR: [4,4]	RN: [1,1] MR: [4,4] PR: [4,4]	STEP: 2 MR: [1.3*2] DR: [2.7*2] LN: [2*2]
$M_{2\times}$	256	192	HC: [48,80,128] RN: [3,3,5] MR: [4,4,4] PR: [4,4,4]	HC: [48,80,128] RN: [3,3,5] MR: [4,4,4] PR: [4,4,4]	RN: [1,1] MR: [4,4] PR: [4,4]	RN: [1,1] MR: [4,4] PR: [4,4]	STEP: 2 MR: [1.3*2] DR: [2.7*2] LN: [2*2]
$M_{4\times}$	384	256	HC: [64,96,192] RN: [3,3,5] MR: [4,4,4] PR: [4,4,4]	HC: [64,96,192] RN: [3,3,5] MR: [4,4,4] PR: [4,4,4]	RN: [1,1] MR: [4,4] PR: [4,4]	RN: [1,1] MR: [4,4] PR: [4,4]	STEP: 2 MR: [1.3*2] DR: [2.7*2] LN: [2*2]
$M_{8\times}$	384	256	HC: [64,128,256] RN: [4,4,7] MR: [4,4,4] PR: [4,4,4]	HC: [64,128,256] RN: [4,4,7] MR: [4,4,4] PR: [4,4,4]	RN: [1,1] MR: [4,4] PR: [4,4]	RN: [1,1] MR: [4,4] PR: [4,4]	STEP: 2 MR: [1.3*2] DR: [2.7*2] LN: [6*2]
$M_{36\times}$	320	256	HC: [96,192,384] RN: [2,2,8] MR: [4,4,4] PR: [4,4,4]	HC: [96,192,384] RN: [2,2,8] MR: [4,4,4] PR: [4,4,4]	RN: [2,2] MR: [4,4] PR: [4,4]	RN: [2,2] MR: [4,4] PR: [4,4]	STEP: 16 MR: [4*16] DR: [1*16] LN: [2*16]

A.5 Local Structure Search

As shown in Figure 9, we decompose the model into four sub-modules. Following Chen et al. (2025), each set of processing layers between adjacent up- and down-sampling layers is treated as one sub-module; since our main model adopts a symmetric design, the encoder and decoder share the same sub-module search space. The entropy model is treated as a separate sub-module.

Our search space is defined as the trade-off between the number of layers and the number of channels within a sub-module. Including the initial base model, we specify five configurations per sub-module, ranging from 1 to 5 layers, and adjust parameters such as the channel count so that the complexity of these five configurations remains roughly comparable.

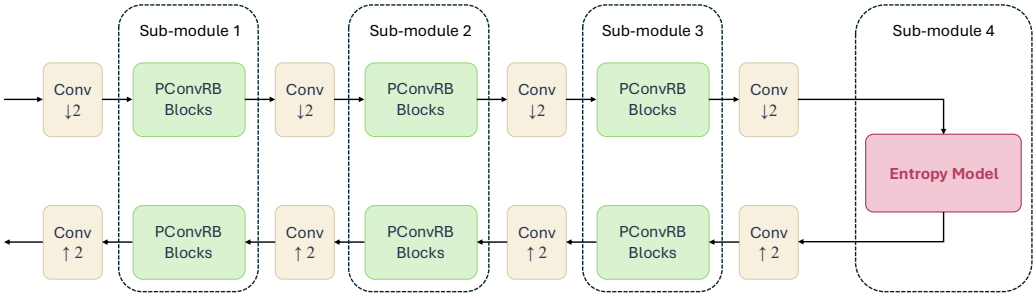


Figure 9: Divide the model into four sub-modules.

B Training Details

B.1 Training Set

CLIC-2020 Training Set. We download the CLIC-2020 training set from the official website [7], which is divided into two categories: professional and mobile. We mix the two categories of datasets and include both in our training set.

DIV2K Training Set. We download the DIV2K training set from the official link Agustsson & Timofte (2017). We only include the high-resolution part in our training set.

Flickr2K Training Set. We download the Flickr2K Timofte et al. (2017) training set from the official link. We only include the high-resolution part in our training set.

We partition all the aforementioned data into complete 512×512 patches, with an overlap of 64 pixels in the overlapping regions, yielding a total of 82,931 image patches as our training set.

B.2 Hierarchical Pruning

Our procedure consists of two complementary phases. We first derive increasingly expressive architectures through gradual model expansion:

$$\mathcal{M}_{1\times} \xrightarrow{\text{expand}} \mathcal{M}_{2\times} \xrightarrow{\text{expand}} \mathcal{M}_{4\times} \xrightarrow{\text{expand}} \mathcal{M}_{8\times}. \tag{16}$$

We then reverse this hierarchy through energy-based hierarchical pruning:

$$\mathcal{M}_{8\times} \xrightarrow{\text{prune+finetune}} \widetilde{\mathcal{M}}_{4\times} \xrightarrow{\text{prune+finetune}} \widetilde{\mathcal{M}}_{2\times} \xrightarrow{\text{prune+finetune}} \widetilde{\mathcal{M}}_{1\times}, \tag{17}$$

where $\mathcal{M}_{k\times}$ denotes the architecture with a decoding complexity of approximately $k \times 20$ kMACs/pixel, and $\widetilde{\mathcal{M}}_{k\times}$ denotes the same target architecture after initialization from the adjacent larger model and subsequent fine-tuning. The expansion path defines structurally compatible models at different complexity levels, while the pruning path transfers the representation learned by the largest model to the compact models.

[7] http://compression.cc/

Starting from the trained $\mathcal{M}_{8\times}$, we construct the compact models through the reverse hierarchy in Equation 17. At each stage, the current model serves as the teacher for initializing the adjacent smaller architecture. Our method jointly reduces predefined network blocks, intermediate feature widths, and latent dimensions.

When a transition reduces the dimensionality of the main latent $\boldsymbol{y}$ or hyper latent $\boldsymbol{z}$, we estimate channel importance from teacher activation energy. Given a calibration set $\mathcal{C}$, the energy of each channel is computed as

$$E_c^y = \frac{\sum_{i\in\mathcal{C}} \sum_{h,w} \left(y_{c,h,w}^{(i)}\right)^2}{\sum_{i\in\mathcal{C}} H_y^{(i)} W_y^{(i)}}, \qquad E_d^z = \frac{\sum_{i\in\mathcal{C}} \sum_{h,w} \left(z_{d,h,w}^{(i)}\right)^2}{\sum_{i\in\mathcal{C}} H_z^{(i)} W_z^{(i)}}, \tag{18}$$

where $\boldsymbol{y}^{(i)}$ and $\boldsymbol{z}^{(i)}$ denote the pre-quantization main latent and hyper latent of image i, respectively. For variable-rate calibration, a quality index is independently sampled for each image unless a fixed operating point is specified. Given target dimensions M' and N', we retain the channels with the highest energies:

$$\mathcal{I}_y = \mathrm{TopK}\left(\{E_c^y\}_{c=1}^{M}, M'\right), \qquad \mathcal{I}_z = \mathrm{TopK}\left(\{E_d^z\}_{d=1}^{N}, N'\right). \tag{19}$$

The selected indices are restored to their original channel order and consistently applied to parameter dimensions corresponding to M and N. Special care is required for modules whose inputs are formed by concatenating two M-channel representations. In our 2-step entropy model, the second spatial-prior takes as input the concatenation of the partially decoded latent and the common hyperprior features. To retain the same channel subset $\mathcal{I}_y$ from both representations, we construct the index sequence

$$\mathcal{I}_{2y} = \mathrm{Concat}\left(\mathcal{I}_y, \mathcal{I}_y + M\right). \tag{20}$$

Here, $+M$ denotes an index offset, rather than an addition to the feature values. This paired indexing preserves the channel correspondence between the partially decoded latent and the hyperprior features after pruning.

The correspondence between network blocks is predetermined by the reverse of gradual expansion. Shared blocks are copied directly, shifted blocks are explicitly remapped, and blocks absent from the student are discarded. For each mapped parameter tensor, we either copy it when the shapes match or slice it to the target shape. Dimensions corresponding to M, N, and $2M$ are sliced using the energy-selected indices, whereas other reduced hidden dimensions retain their leading channels. Parameters without a valid correspondence keep their default student initialization.

After parameter transfer, the entire student is fine-tuned using the original rate-distortion objective. The teacher is used only for energy estimation and parameter initialization and is removed before

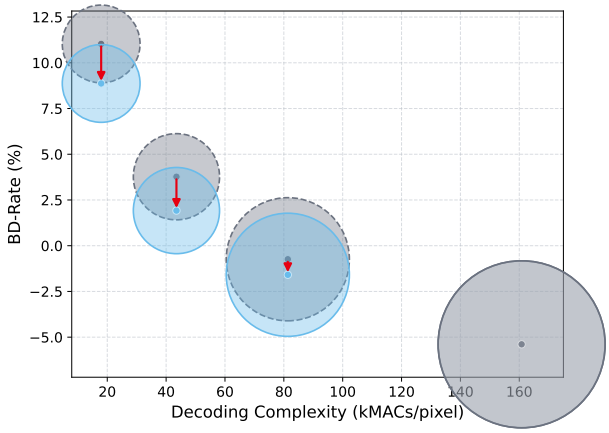


Figure 10: Complexity-performance trade-off before and after pruning. Solid blue bubbles denote the pruned models, dashed gray bubbles denote their unpruned counterparts, and red arrows indicate the BD-rate gains from pruning.

fine-tuning; therefore, no distillation loss or auxiliary teacher branch is required. The converged student then becomes the teacher for the next stage.

C Evaluation Details

C.1 Traditional Codecs

HM. We use HM-18.0 [8] with the All-Intra configuration `encoder_intra_main_rext.cfg`. RGB images are converted to 8-bit YUV 4:4:4 format:

```
TAppEncoder -c encoder_intra_main_rext.cfg -i input.yuv \
  -b output.bin -o recon.yuv -wdt <W> -hgt <H> \
  -q <QP> -f 1 -fr 1 \
  --InputBitDepth=8 --InternalBitDepth=8 \
  --InputChromaFormat=444
```

VTM. We use VTM-24.0 [9] with the All-Intra configuration `encoder_intra_vtm.cfg`. RGB images are converted to 8-bit YUV 4:4:4 format:

```
EncoderApp -c encoder_intra_vtm.cfg -i input.yuv \
  -b output.bin -o recon.yuv -wdt <W> -hgt <H> \
  -q <QP> -f 1 -fr 1 \
  --InputBitDepth=8 --InternalBitDepth=8 \
  --InputChromaFormat=444 \
  --TemporalSubsampleRatio=1 \
  --ConformanceWindowMode=1
```

C.2 Learned Codecs

ELIC (He et al., 2022). For the base version, we evaluate the compression performance using a third-party reimplementation [10]; however, for ELIC-SM, since no pretrained weights are available, we only report the numbers provided in the original paper.

EVC (Wang et al., 2023). We use the officially released code [11] and pretrained weights.

JPEG-AI (Ascenso et al., 2023). We use the officially released code [12] and pretrained weights.

ShiftLIC (Bao et al., 2025) We use the officially released code [13] to validate the computational complexity. Since the pretrained weights have not been released, we instead adopt the results reported in the official paper.

HyperPrior (Ballé et al., 2018). We conduct the evaluation using the code and pretrained weights from a third-party implementation [14].

Shallow-NTC (Yang & Mandt, 2023). We use the officially released code [15] and pretrained weights.

FastNIC (Zhang et al., 2025b). The model is not open-sourced; after contacting the authors, we obtained the raw data from them.

[8] https://vcgit.hhi.fraunhofer.de/jct-vc/HM
[9] https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM
[10] https://github.com/VincentChandelier/ELiC-ReImplemetation
[11] https://github.com/microsoft/DCVC/tree/main/DCVC-family/EVC
[12] https://gitlab.com/wg1/jpeg-ai/jpeg-ai-reference-software
[13] https://github.com/baoyu2020/ShiftLIC
[14] https://github.com/InterDigitalInc/CompressAI
[15] https://github.com/mandt-lab/shallow-ntc

He2026 (He et al., 2026). The model is not open-sourced; after contacting the authors, we obtained the raw data from them.

MLIC++ (Jiang et al., 2025). We use the officially released code [16] and pretrained weights.

FLIC (Li et al., 2024). We use the officially released code [17] and pretrained weights.

LALIC (Feng et al., 2025). We use the officially released code [18] and pretrained weights.

HPCM (Li et al., 2025). We use the officially released code [19] and pretrained weights.

D More Results

D.1 Comparison on More Datasets

We further evaluate our model on the Tecnick dataset (Giachetti, 2014) (100 images at 1200×1200), and the detailed experimental results are as follows:

Table 5: Rate-distortion performance and complexity comparison on the Tecnick dataset. The anchor is VTM-24.0.

Method	**BD-rate**(%)	**Method**	**BD-rate**(%)	**Method**	**BD-rate**(%)
HM-18.0	+29.03	HyperPrior	+39.35	Shallow-NTC	+42.12
EVC-S	+8.52	EVC-M	+1.52	EVC-L	-1.15
Shift-LIC-Small	+18.36	Shift-LIC-Middle	+11.15	Shift-LIC-Large	-1.00
ELIC	-5.21	FastNIC	+29.56	MLIC++	-15.89
FLIC	-17.15	LALIC	-19.92	HPCM	-18.06
$\mathcal{M}_{0.25\times}$(Ours)	+35.79	$\mathcal{M}_{0.5\times}$(Ours)	+27.79	$\mathcal{M}_{1\times}$(Ours)	+11.05
$\mathcal{M}_{2\times}$(Ours)	+0.87	$\mathcal{M}_{4\times}$(Ours)	-4.08	$\mathcal{M}_{8\times}$(Ours)	-9.27
$\mathcal{M}_{36\times}$(Ours)	-15.22	$\mathcal{M}_{1\times}^{2\times}$(Ours)	+9.71		

[16] https://github.com/JiangWeibeta/MLIC
[17] https://github.com/qingshi9974/ICLR2024-FTIC
[18] https://github.com/sjtu-medialab/RwkvCompress
[19] https://github.com/lyq133/LIC-HPCM

D.2 Complete Rate-distortion Curve and More Visualization

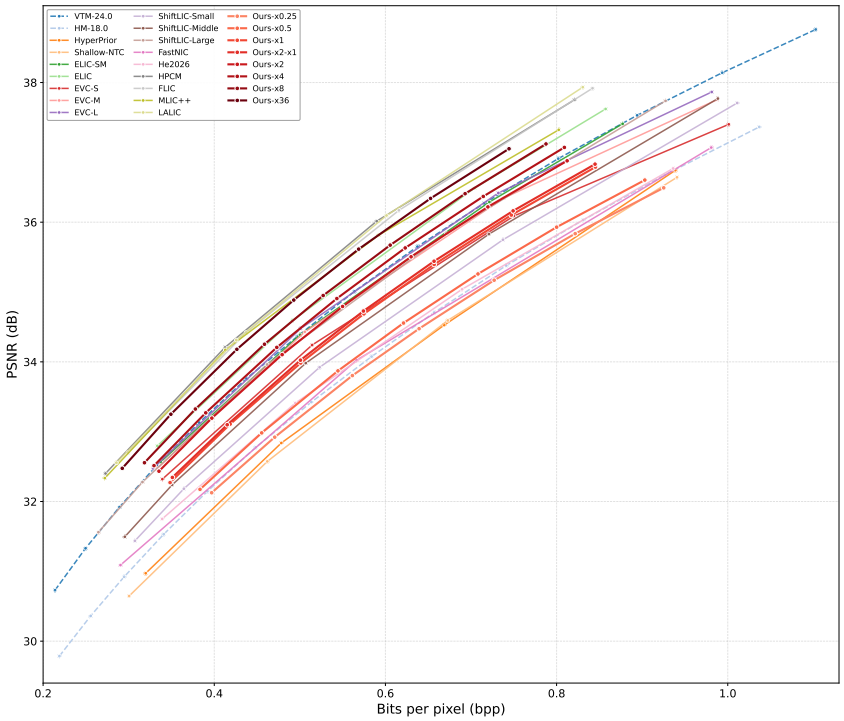


Figure 11: Rate-distortion curve on Kodak.

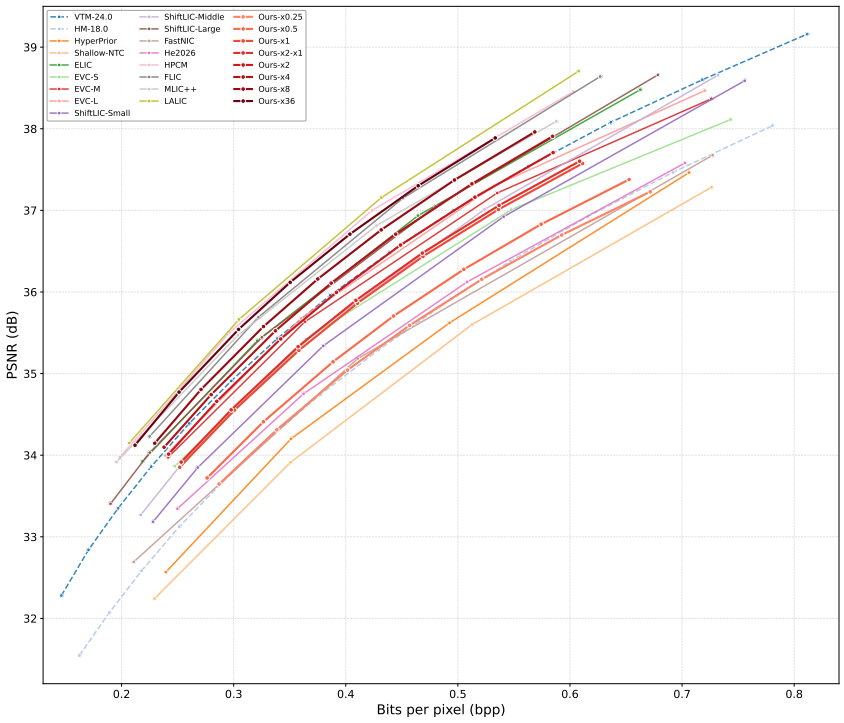


Figure 12: Rate-distortion curve on CLIC-PV.



Figure 13: Visual comparison on kodim24.



Figure 14: Visual comparison on daniel-robert-405.

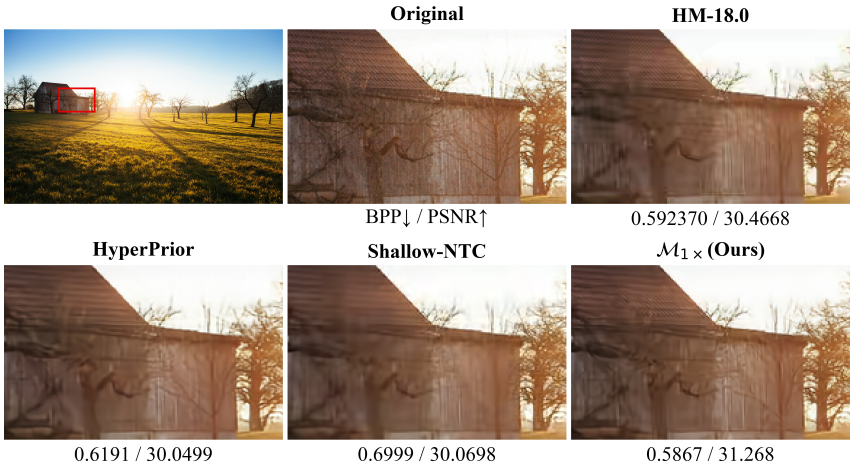


Figure 15: Visual comparison on philipp-reiner-207.