SWA-IC: MULTIRATE LEARNED IMAGE COMPRESSION WITH SLIDING WINDOW ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Learned image compression has surpassed conventional codecs in rate–distortion performance, but many leading methods require a separately trained model for each operating point. Supporting multiple rates therefore increases training and storage requirements and complicates deployment. We present SWA-IC, a single image compression model supporting 128 fine-grained operating points with state-of-the-art rate–distortion performance. We extend sliding window attention from autoregressive context modeling to the analysis and synthesis transforms, establishing a common SWA-based design for feature transformation and spatial context modeling. We give the synthesis transform direct access to context features for image reconstruction and equip SWA with learned attention sinks to suppress uninformative contributions. Growing channel groups and generalized Gaussian entropy modeling strengthen the parallel context model, while channel-wise gains control rate with shared network weights across all operating points. On Kodak, Tecnick, and CLIC, SWA-IC achieves BD-rate savings of 18.44%, 24.86%, and 18.53%, respectively, relative to VTM 23.10, obtaining the lowest BD-rate among the evaluated single-rate and multirate codecs. At medium and large tested resolutions, it also matches or improves on the decoding latency of previous state-of-the-art image codecs.

1 INTRODUCTION

Learned image compression has advanced through improvements to both feature transforms and latent probability estimation. Recent methods combine convolution and transformers (Liu et al., 2023), introduce linear attention (Feng et al., 2025), or enrich entropy models with learned dictionaries (Lu et al., 2025), achieving rate–distortion performance beyond conventional codecs. However, many previous state-of-the-art image codecs require a separately trained model for each operating point. Deploying these methods across multiple rates therefore multiplies training and storage requirements, limiting their deployment flexibility.

Multirate compression addresses this limitation by supporting multiple operating points within one model, but existing approaches make different compromises. DCVC-RT (Jia et al., 2025) combines gain-based rate adaptation with a real-time coding architecture, but its intra model remains behind leading image codecs in rate–distortion performance. MixCompress (Ta et al., 2026) pursues stronger compression with a shared backbone and rate-dependent experts, but storing multiple sets of expert weights still adds substantial storage overhead to its backbone despite avoiding separate models for each rate. This raises the question: Can a single model with shared network weights across all rates achieve state-of-the-art rate–distortion performance and competitive decoding speed?

We address this question with SWA-IC, a multirate image codec built around sliding window attention (SWA). Starting from P-SWA's parallel context-modeling framework (Kopte & Kaup, 2026), we extend SWA (Kopte & Kaup, 2025) to both analysis and synthesis transforms, establishing a common SWA-based design for feature transformation and spatial context modeling. We give the synthesis transform direct access to the context features alongside the decoded latent, enabling it to reuse these features for image reconstruction. Within this design, we equip SWA with learned attention sinks so that attention heads can suppress uninformative contributions. To improve P-SWA's entropy modeling, we introduce growing channel groups and replace its Gaussian distributions with generalized Gaussian distributions (GGDs), allowing the predicted distribution shapes to adapt to

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.

the latents. We train the resulting codec across operating points with shared network weights, using channel-wise gains for rate control.

SWA-IC supports 128 operating points and achieves the lowest BD-rate among the evaluated single-rate and multirate codecs on Kodak, Tecnick, and CLIC. It also matches or improves on the decoding latency of previous state-of-the-art image codecs at medium and large resolutions.

Our main contributions are:

- We extend SWA to the analysis and synthesis transforms and incorporate learned attention sinks. We give the nonlinear synthesis branch and its wavelet shortcut direct access to context features.
- We improve P-SWA's context model with growing channel groups and generalized Gaussian entropy modeling while retaining parallel decoding.
- We demonstrate state-of-the-art rate–distortion performance across three datasets with one model supporting 128 gain-controlled operating points and competitive decoding speed.

2 Related Work

Learned image compression. Learned image codecs jointly optimize feature transforms and entropy models under a rate–distortion objective (Ballé et al., 2017). A central direction of progress has been to make entropy models better exploit latent dependencies. Hyperpriors provide side information for predicting latent distributions (Ballé et al., 2018), while spatial and channel autoregressive models further improve these predictions using previously decoded elements (Minnen et al., 2018; Minnen & Singh, 2020). Checkerboard context modeling enables parallel decoding within spatial groups (He et al., 2021), and DCVC-DC extends this approach with a quadtree partition (Li et al., 2023). Later work explores richer context and more structured decoding schedules: MLIC++ (Jiang et al., 2025) combines channel-wise, local spatial, and global spatial context, while HPCM (Li et al., 2025b) progressively incorporates context through a hierarchical coding schedule. Complementing these advances in context modeling, Zhang et al. (2025) replace Gaussian latent distributions with generalized Gaussian distributions, allowing their shape to be learned. Beyond entropy modeling, GLIC (Chen et al., 2026a) uses graph neural networks with content-adaptive connectivity to model spatial redundancy. SWA-IC builds on spatial and channel context modeling, combining P-SWA's parallel decoding schedule (Kopte & Kaup, 2026) with growing channel groups and generalized Gaussian probability estimates.

Attention and state-space models. Attention has been explored in entropy models, as in Entroformer (Qian et al., 2022) and Contextformer (Koyuncu et al., 2022), and in feature transforms (Zhu et al., 2022; Zou et al., 2022; Liu et al., 2023). Its computational cost depends strongly on how spatial interactions are modeled. STF uses local Swin attention (Liu et al., 2021), while TCM combines window-based attention with convolution. Window partitioning limits complexity but produces asymmetric neighborhoods that depend on each query's position within its window. M2T (Mentzer et al., 2023) accelerates entropy decoding through grouped prediction within independently coded latent patches, but its context remains confined to each patch. LALIC (Feng et al., 2025) and CMIC (Chen et al., 2026b) instead use linear attention and state-space modeling to capture long-range dependencies without the quadratic cost of global softmax attention. We retain softmax attention while limiting its computation to a local window centered on each query. This avoids the asymmetric neighborhoods caused by window partitioning, with complexity growing linearly in the number of spatial positions for a fixed window size. SWA (Kopte & Kaup, 2025) introduced sliding window attention for video context modeling, and P-SWA (Kopte & Kaup, 2026) extended it to parallel decoding. We extend it to both image transforms and adapt P-SWA's context model to image compression. Our transforms follow DCVC-RT's non-cascaded structure without intermediate downsampling (Jia et al., 2025).

Multirate compression. Multirate codecs differ in whether they adapt intermediate features, quantization, or network structure. Conditional autoencoders (Choi et al., 2019), feature modulation (Yang et al., 2020), and quality-controlled spatial feature transforms (Song et al., 2021) condition intermediate representations on the desired operating point. QARV (Duan et al., 2024) uses rate-conditioned adaptive layer normalization, following the condition-dependent affine formulation of

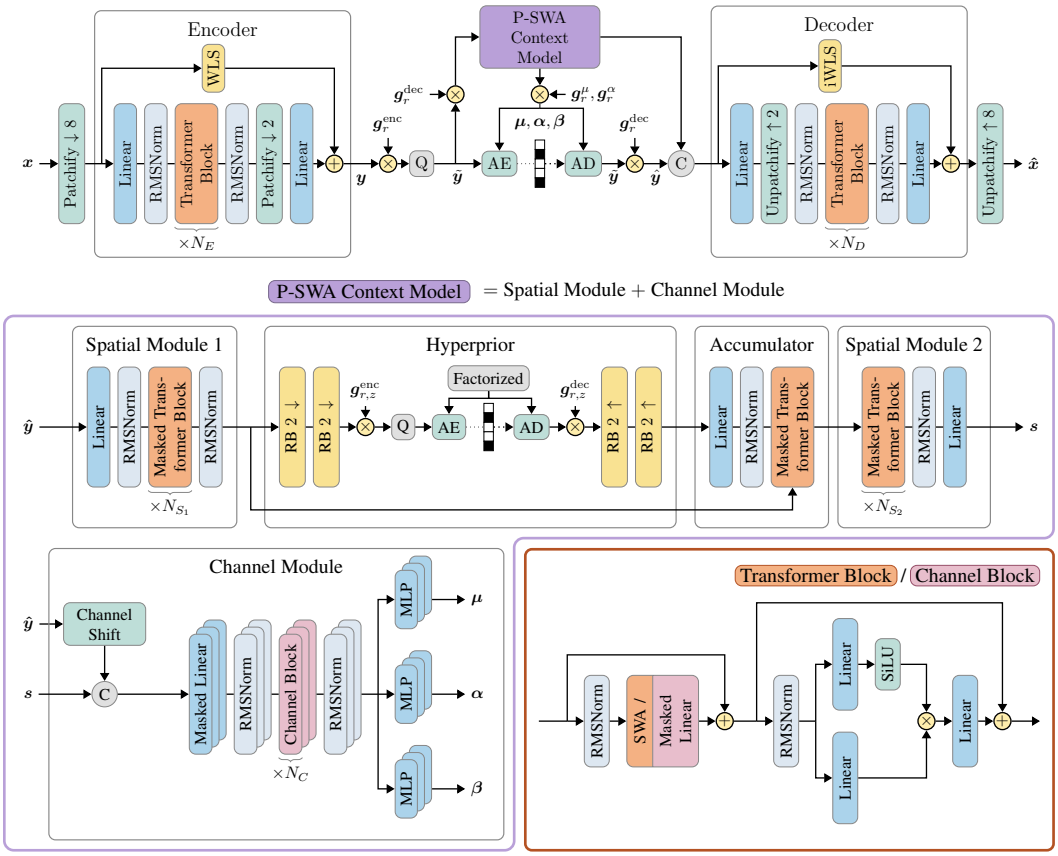

Figure 1: SWA-IC architecture and block structure. WLS/iWLS: analysis/synthesis wavelet-based linear shortcuts; RB: convolutional residual block; Q: quantization; AE/AD: arithmetic encoding/decoding; C: concatenation.

dynamic layer normalization (Kim et al., 2017). Gained autoencoders instead learn channel-wise scaling vectors and interpolate between them (Cui et al., 2021), while STanH learns parametric quantizers for different rates (Presta et al., 2025). MixCompress (Ta et al., 2026) introduces rate-dependent expert selection and feature modulation, with its mixture-of-depths variant also adapting processing depth. We adopt gain-based adaptation outside the transformer blocks to support fine-grained rate control while sharing the network weights across all operating points.

3 Method

3.1 Architecture

Our architecture (Figure 1) uses 2D Sliding Window Attention (SWA) (Section 3.2) for both feature transformation and context modeling. Following DCVC-RT (Jia et al., 2025), we reduce spatial resolution before the transformer stack so that most feature processing operates on low-resolution representations. Our transformer blocks use pre-normalized residual SWA and SwiGLU sublayers following Touvron et al. (2023), with RMSNorm (Zhang & Sennrich, 2019). Both transforms include independently learned wavelet-based linear shortcuts (Li et al., 2025a) to transmit DC components and stabilize early training.

Analysis transform. The analysis transform flattens non-overlapping 8×8 patches of the input image $\boldsymbol{x} \in \mathbb{R}^{3\times H\times W}$ into 192-dimensional vectors. The nonlinear branch applies a linear embedding, RMSNorm, N_E SWA transformer blocks, RMSNorm, 2×2 patch merging, and a linear projection. Adding the shortcut output gives latent $\boldsymbol{y} \in \mathbb{R}^{C\times \frac{H}{16}\times \frac{W}{16}}$, with $C = 256$.

Rate adaptation. Our analysis transform, synthesis transform, and context model share their weights across all operating points. Rate-dependent gains control latent scaling outside the transformer blocks because normalization within the blocks could counteract their effect. For primary latent $\boldsymbol{y}$ and its entropy parameters, four independently learned banks of channel-wise gain vectors support $P = 128$ discrete operating points. For rate point r, encoder and decoder gains produce the transmitted quantized latent $\tilde{\boldsymbol{y}}$ and its decoder-scaled representation $\hat{\boldsymbol{y}}$:

$$\tilde{\boldsymbol{y}} = \mathrm{round}(\boldsymbol{g}_r^{\mathrm{enc}} \odot \boldsymbol{y}), \qquad \hat{\boldsymbol{y}} = \boldsymbol{g}_r^{\mathrm{dec}} \odot \tilde{\boldsymbol{y}}. \tag{1}$$

During training, rounding uses a straight-through estimator (STE) for gradients. The context model and synthesis transform receive the decoder-scaled latent $\hat{\boldsymbol{y}}$, while symbol probabilities are computed for the transmitted symbols $\tilde{\boldsymbol{y}}$. Gain banks $\boldsymbol{g}_r^{\mu}$ and $\boldsymbol{g}_r^{\alpha}$ therefore adjust the predicted distribution's location and scale to account for this difference in scaling. Two additional, independently learned banks scale hyperlatent $\boldsymbol{z}$ before quantization and after decoding.

Spatial context and hyperprior. We remove the temporal components of P-SWA (Kopte & Kaup, 2026) and use the decoder-scaled latent $\hat{\boldsymbol{y}}$ as input to Spatial Module 1. The convolutional hyperprior compresses the resulting features into side information $\tilde{\boldsymbol{z}}$, using a separate factorized prior for each rate point as in DCVC-RT (Jia et al., 2025). The accumulator combines lossy side information from the decoded hyperprior with losslessly decoded spatial context from previously decoded neighbors through strictly masked cross-attention. Spatial Module 2 processes the accumulator output before passing it to the channel module. Both spatial modules use non-strict masking; Section 3.3 explains how the combined masks preserve causality.

Channel context. Inspired by ELIC's uneven channel grouping (He et al., 2022), we use small initial channel groups to provide context for larger subsequent groups, retaining only three channel-decoding stages per spatial step. We partition the 256-channel latent into groups of widths $(28, 56, 172)$. Each group-specific branch receives the output of Spatial Module 2 and symbols from previously decoded channel groups. Following P-SWA, its residual blocks replace SWA with masked linear layers that mix hidden features from the current and preceding branches. Each input embedding excludes its own undecoded symbols and all future groups. Group-specific heads predict entropy parameters from each branch's final normalized output. Each branch uses a hidden width four times its latent-group width. For synthesis, we concatenate the branch outputs into $\boldsymbol{H}_{\mathrm{ctx}}$, yielding $4 \times 256 = 1024$ channels.

Entropy model. A Gaussian entropy model adapts location and scale but fixes the distribution shape. Building on the generalized Gaussian entropy modeling of Zhang et al. (2025), we extend our context-model heads to predict location, scale, and shape at each spatial position and channel. We allow shapes from Laplacian ($\beta = 1$) through Gaussian ($\beta = 2$) to increasingly uniform-like distributions, with $\beta \in [1, 7]$. The rate-dependent gains adjust location and scale, while shape is not rate-scaled.

Synthesis transform. P-SWA uses Latent Residual Prediction (LRP) to refine the decoded latent, but this destabilized joint training in our model. In that configuration, the LRP head was trained through the overall rate–distortion loss, without a separate loss for predicting quantization error or a constraint on its output range. We hypothesize that it could therefore learn part of the synthesis mapping outside the stabilizing wavelet shortcut, contributing to the instability. Instead, we give the synthesis transform direct access to the context features alongside the decoded latent. Both the nonlinear branch and its linear shortcut receive

$$\boldsymbol{H}_{\mathrm{syn}} = \mathrm{Concat}(\hat{\boldsymbol{y}}, \boldsymbol{H}_{\mathrm{ctx}}) \in \mathbb{R}^{1280 \times \frac{H}{16} \times \frac{W}{16}}. \tag{2}$$

The nonlinear branch applies a linear 2×2 upsampling embedding, RMSNorm, N_D transformer blocks, RMSNorm, and a linear output projection. We add the nonlinear branch's output to the shortcut output and rearrange the resulting patches into the reconstructed image $\hat{\boldsymbol{x}} \in \mathbb{R}^{3 \times H \times W}$.

3.2 Sliding Window Attention

SWA (Kopte & Kaup, 2025) aggregates features within a local window centered on each query. Like standard convolution, SWA uses the same local window geometry at every position. Its attention weights, however, adapt to the input rather than being fixed after training.

These adaptive weights are computed from query and key features. Writing the spatial features as rows of $X \in \mathbb{R}^{HW \times C}$, each self-attention head i computes the standard projections (Vaswani et al., 2017)

$$Q^{(i)} = XW_Q^{(i)}, \qquad K^{(i)} = XW_K^{(i)}, \qquad V^{(i)} = XW_V^{(i)}, \tag{3}$$

where $W_Q^{(i)}$, $W_K^{(i)}$, and $W_V^{(i)}$ are learned projections. In the accumulator's cross-attention, queries instead come from the hyperprior features, while keys and values come from Spatial Module 1. Let $\boldsymbol{q}_m^{(i)}$, $\boldsymbol{k}_n^{(i)}$, and $\boldsymbol{v}_n^{(i)}$ denote the projected query, key, and value vectors at spatial positions m and n.

Each query at position m attends only to positions in its local neighborhood $\mathcal{N}(m)$. In the feature transforms, this includes all image positions within the window. In the context model, the prediction mask additionally restricts which positions are available (Section 3.3).

Standard softmax attention assigns weights that sum to one across the available values, even when none is useful. Following GPT-OSS (OpenAI, 2025) and related off-by-one attention (Miller, 2023), we add a learned sink to the softmax denominator, allowing the weights on these values to sum to less than one. This lets the head reduce its output. Each head has a learned sink logit $s_{\text{sink}}^{(i)}$, initialized to zero. The attention logits and output are

$$\begin{aligned} \ell_{mn}^{(i)} &= \frac{(\boldsymbol{q}_m^{(i)})^\top \boldsymbol{k}_n^{(i)}}{\sqrt{d_k}} + b^{(i)}_{\Delta y_{mn}, \Delta x_{mn}}, \\ \boldsymbol{a}_m^{(i)} &= \frac{\sum_{n \in \mathcal{N}(m)} \exp(\ell_{mn}^{(i)})\, \boldsymbol{v}_n^{(i)}}{\exp(s_{\text{sink}}^{(i)}) + \sum_{n \in \mathcal{N}(m)} \exp(\ell_{mn}^{(i)})}. \end{aligned} \tag{4}$$

Here, d_k is the query/key dimension and $b^{(i)}$ contains learned relative-position biases. The displacements Δy_{mn} and Δx_{mn} denote the vertical and horizontal offsets from query position m to key position n. Stacking $(\boldsymbol{a}_m^{(i)})^\top$ as rows gives $A^{(i)}$; the h head outputs are concatenated and projected:

$$O = \text{Concat}(A^{(1)}, \ldots, A^{(h)})W_O. \tag{5}$$

We implement 2D SWA in Triton (Tillet et al., 2019) using a kernel based on FlashAttention-2 (Dao, 2024).

3.3 Prediction Scheme

We retain P-SWA's four-step diagonal pattern (Figure 2). At each spatial step, the three channel groups are decoded sequentially, with all positions of the same step decoded in parallel within each group. Each group becomes available to the next; decoding advances to the next spatial step after all three groups are complete. Latent y thus requires $4 \times 3 = 12$ sequential stages regardless of image resolution, preceded by hyperlatent decoding.



Figure 2: Four-step diagonal decoding order, repeated spatially.

To decode in this order, each prediction must exclude symbols that are not yet available. P-SWA's masking scheme enforces this through the two spatial modules and the accumulator. Let τ denote the spatial prediction step and consider the output of Spatial Module 2 at position n. Its non-strict mask admits accumulator outputs at positions m from the same or earlier steps. At each m, the accumulator's strict cross-attention mask restricts keys and values to Spatial Module 1 outputs at positions p from strictly earlier steps. Spatial Module 1's non-strict mask, in turn, allows these outputs to depend on input positions q from the same or earlier steps as p. Together, the masks enforce

$$\tau(q) \le \tau(p) < \tau(m) \le \tau(n). \tag{6}$$

The strict inequality introduced by the accumulator ensures $\tau(q) < \tau(n)$, so only symbols from earlier prediction steps contribute through this path, even though both spatial modules use non-strict masks. The hyperprior provides a separate path that may depend on current and future positions because $\tilde{\boldsymbol{z}}$ is decoded first.

3.4 JOINT MULTIRATE TRAINING

We train all 128 operating points jointly, using a separate rate–distortion multiplier λ_r for each point. We geometrically space the multipliers as

$$\lambda_r = 100 \left(\frac{5000}{100} \right)^{r/(P-1)}, \qquad r = 0, \dots, P-1, \quad P = 128. \tag{7}$$

Each batch contains one image per operating point ($b = 128$), so every update trains the full operating range.

The context model and synthesis transform receive rounded latents, with straight-through gradients. To obtain differentiable rate estimates, we instead perturb the encoder-scaled latents with uniform noise for the rate loss, yielding $\tilde{\boldsymbol{y}}^{\text{noise}}$ and $\tilde{\boldsymbol{z}}^{\text{noise}}$. For rate indices r_n, we minimize

$$L = \frac{1}{b} \sum_{n=1}^{b} \left[R(\tilde{\boldsymbol{y}}_n^{\text{noise}}) + R(\tilde{\boldsymbol{z}}_n^{\text{noise}}) + \lambda_{r_n} D(\boldsymbol{x}_n, \hat{\boldsymbol{x}}_n) \right] + 0.1 L_{\text{orth}}. \tag{8}$$

Here, R is the summed negative log-probability per source-image pixel, using the GGD model for y and the factorized prior for z. Distortion D is RGB mean squared error for targets in $[0, 1]$. Following Li et al. (2025a), L_{orth} regularizes the analysis and synthesis shortcut projections toward orthogonality. Additional architecture and training details are provided in Appendices A.1 and A.2.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETTING

Training. We train SWA-IC end-to-end on the whole winter 2021 ImageNet dataset (Deng et al., 2009) using 256×256 crops and AdamW. Training runs for 12 epochs (approximately $1.2\,\text{M}$ iterations), with a cosine learning-rate schedule from 2×10^{-5} to 1×10^{-6}. Training uses automatic mixed precision (AMP) and takes approximately 164–165 hours on one NVIDIA H100. We evaluate the final checkpoint without parameter averaging or validation-based selection.

Evaluation. We test on Kodak (Franzen, 1999), Tecnick (Asuni & Giachetti, 2015), and CLIC Pro validation (Toderici et al., 2020). For SWA-IC, DCVC-RT-Intra, and VTM 23.10 (JVET, 2025), we run our own rate–distortion evaluations, measuring PSNR in RGB after clipping reconstructions to $[0, 1]$ and obtaining bitrate from the produced bitstreams. DCVC-RT-Intra runs in native FP16, while VTM uses the All-Intra configuration with YUV 4:4:4 input. For the single-rate baselines, we use published FP32 rate–distortion results from their official repositories. For MixCompress-MoD, we compute BD-rates from numerical rate–distortion points provided directly by its authors.

Runtime measurement. We measure runtime on an NVIDIA RTX 3090 using AMP, except for DCVC-RT-Intra, which uses native FP16, and TCM's entropy-coding path, which retains FP32. We disable DCVC-RT's optional CUDA inference extensions to avoid optimizations not applied across codecs, such as operator fusion and reduced Python and dispatch overhead. Our SWA kernel is integrated into PyTorch and therefore follows its standard dispatch path; it does not fuse surrounding layers. All reported timings are measured using the evaluated implementations and checkpoints under this protocol. Appendix A.4 details and validates this protocol.

4.2 RESULTS

4.2.1 RATE–DISTORTION PERFORMANCE AND RUNTIME

With one model supporting 128 operating points, SWA-IC achieves the lowest BD-rate among all evaluated single-rate and multirate codecs on Kodak, Tecnick, and CLIC (Table 1). Its BD-rate reductions relative to VTM are 18.44%, 24.86%, and 18.53%, respectively. The strongest single-rate competitors differ across datasets: HPCM Large leads the baselines on Kodak and CLIC, while CMIC performs best on Tecnick. Direct comparisons over each pair's full overlapping PSNR interval show bitrate savings of 5.00%, 5.04%, and 3.05% over MixCompress-MoD, achieved with

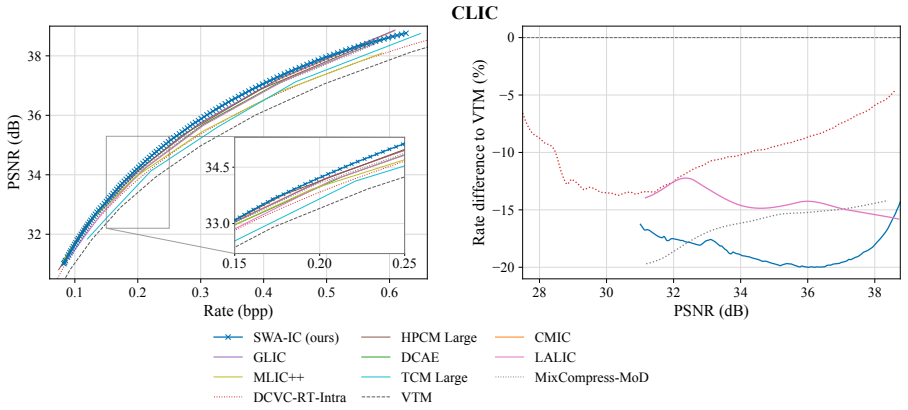

Figure 3: Rate–distortion curves on CLIC (left; dotted lines denote multirate baselines) and local bitrate differences relative to VTM 23.10 (right). Local differences span each model's full measured quality overlap with VTM.

Table 1: BD-rate relative to VTM 23.10 over each codec's overlapping PSNR interval; intervals therefore differ between codecs. Decoding kMACs/px are measured at 2048×2048; timings use an RTX 3090 and include arithmetic coding, which MAC counts exclude.

	BD-rate (%) ↓			Decoding	Decoding time (s) ↓		
Method	**Kodak**	**Tecnick**	**CLIC**	**kMACs/px**	768×512	1200×1200	2048×2048
Single-rate models							
TCM Large (Liu et al., 2023)	−10.16	−11.44	−8.04	1153.3	0.163	0.453	1.060
MLIC++ (Jiang et al., 2025)	−13.39	−17.61	−13.11	777.8	0.202	0.385	0.801
DCAE (Lu et al., 2025)	−15.41	−20.57	−15.81	642.6	0.115	0.341	0.851
GLIC (Chen et al., 2026a)	−17.74	−22.36	−17.81	726.3	0.186	0.568	1.418
CMIC (Chen et al., 2026b)	−16.42	−23.52	−17.91	583.7	0.164	0.399	1.056
HPCM Large (Li et al., 2025b)	−18.43	−21.83	−18.11	1006.8	0.058	0.176	0.412
LALIC (Feng et al., 2025)	−13.69	−18.10	−14.27	377.8	0.181	0.507	1.246
Multirate models							
DCVC-RT-Intra (Jia et al., 2025)	−12.31	−15.99	−9.80	**362.2**	**0.021**	**0.057**	**0.141**
MixCompress-MoD‡ (Ta et al., 2026)	−14.44	−21.14	−16.18	—	—	—	—
SWA-IC (Ours)	**−18.44**	**−24.86**	**−18.53**	1219.6	0.106	0.176	0.362

‡ MixCompress-MoD reports slightly slower decoding than LALIC; code and checkpoints are unavailable.

channel-wise gains rather than rate-dependent expert selection or processing depth. The corresponding savings over DCVC-RT-Intra are 10.03%, 12.85%, and 10.02%. SWA-IC's advantage over both single-rate and multirate baselines also persists when all codecs are evaluated over a common quality interval (Appendix Table 7).

SWA-IC's 128 operating points cover approximately 0.10–0.86 bpp on Kodak, 0.09–0.58 bpp on Tecnick, and 0.08–0.63 bpp on CLIC. DCVC-RT-Intra extends to lower bitrates than SWA-IC, which targets the range of the evaluated state-of-the-art single-rate codecs. On CLIC, SWA-IC and DCVC-RT-Intra achieve their largest savings relative to VTM in the middle of their operating ranges, with smaller savings toward both ends (Figure 3). MixCompress's savings relative to VTM also decrease toward higher rates. Since MixCompress builds on the single-rate codec LALIC, we compare their curves to examine whether this decline is inherited from its backbone. LALIC instead stabilizes toward high rates on Kodak and generally improves on Tecnick and CLIC. MixCompress's advantage over LALIC is concentrated at low rates and reverses at high rates on all three datasets (Appendix Figure 9). Thus, despite its more elaborate rate adaptation, MixCompress also exhibits the high-rate decline in relative performance observed for SWA-IC and DCVC-RT-Intra.

Compared with the evaluated single-rate codecs, SWA-IC has the lowest decoding latency at 2048×2048, ties HPCM Large for first at 1200×1200 at the reported precision, and ranks second at 768×512 (Table 1). P-SWA's masked decoding (Kopte & Kaup, 2026) requires advanced indexing and non-contiguous memory accesses. These costs account for a larger share of runtime

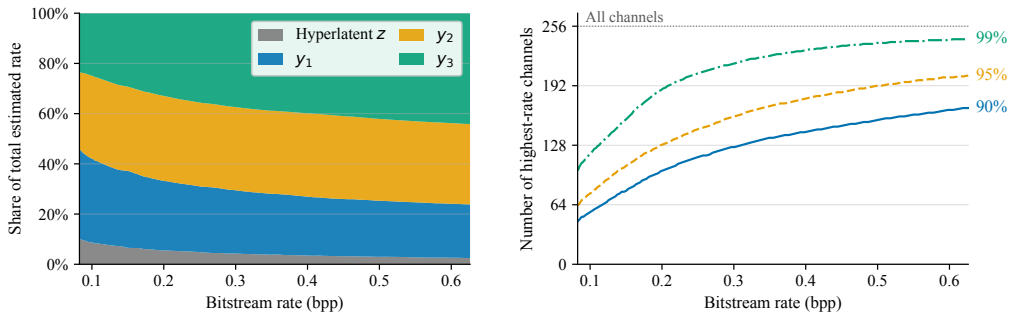

Figure 4: Rate allocation on CLIC. Left: component shares of total estimated rate. Right: minimum number of channels accounting for 90%, 95%, and 99% of latent y rate, ranked by estimated bits pooled across the dataset.

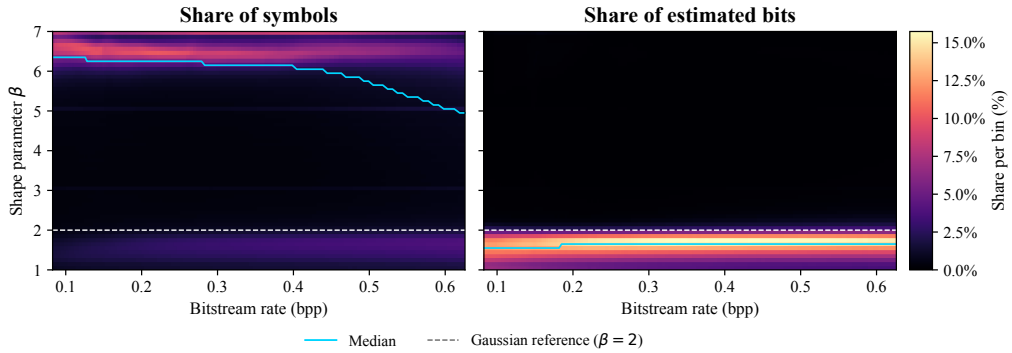

Figure 5: GGD shape distributions across all channels on CLIC. Color shows the share of symbols (left) or estimated bits (right) per β bin, using a shared scale. Each vertical distribution sums to 100%. Dashed lines mark $\beta = 2$; cyan curves show medians.

on small images, explaining why SWA-IC's decoding advantage increases with resolution. We cannot time MixCompress-MoD under our protocol because its code and checkpoints are unavailable. However, it adds components to LALIC and reports slightly slower decoding. LALIC's measured decoding times of 1.7–3.4× ours therefore strongly support a substantial decoding advantage over MixCompress-MoD. These bitrate savings, however, come at higher decoding cost than DCVC-RT-Intra: at 2048 × 2048, SWA-IC requires 3.37× the decoding MACs and 2.57× the decoding time.

4.2.2 Latent Analysis

We examine how the learned rate allocation and distribution shapes relate to our channel grouping and entropy-model design on CLIC. Figure 4 (left) shows that, as bitrate increases, the hyperlatent's share decreases while y_3's share increases. Despite their unequal widths, the three channel groups account for comparable shares of the estimated rate, so earlier groups carry considerably more rate per channel, particularly at low rates.

This redistribution accompanies increasing channel activity (Figure 4, right). At each operating point, we rank channels by their estimated bits summed across the dataset and count those needed to reach each rate fraction. Here, inactive denotes channels or elements with negligible estimated rate. Around 0.25–0.30 bpp, the 99% curve begins to flatten, while the 90% and 95% curves continue rising with decreasing slopes. This suggests that most channels contribute across the dataset by the middle of the bitrate range and that further rate increases spread information more evenly across them. A small tail persists: even at the highest operating point, some channels jointly contribute less than 1% of latent y rate. These may represent specialized features with limited average contributions.

We compare the frequency of predicted distribution shapes with their share of estimated rate. At each operating point, we pool shape parameters across all channels, spatial positions, and CLIC images into histograms, weighting elements equally or by estimated code length (Figure 5). Most predicted shape parameters lie in two ranges, $\beta \in [1, 2]$ and $\beta \in [6, 7]$, with few elements between these ranges. Weighting by estimated code length concentrates the distribution almost entirely in the first range: rate-carrying elements favor Laplacian-to-Gaussian shapes, whereas inactive elements favor uniform-like shapes. The scarcity of intermediate values suggests that they are less useful for either role in this model.

4.3 Ablation Studies

To assess each component's contribution, we trained ablations from scratch, changing only the ablated component, and evaluated them with identical coding settings on CLIC. Table 2 reports BD-rate relative to our full model over pairwise overlapping PSNR intervals; positive values indicate worse performance.

Table 2: Ablations on CLIC relative to the full SWA-IC model.

Modification	**ΔBD-rate (%)**
Full model	0.00
No attention sinks	+1.07
No SWA in transforms	+1.71
No GGD	+3.47
No context features for synthesis	+8.53
No growing channel groups	+9.62

Attention sinks. Removing attention sinks increases BD-rate by 1.07%. We hypothesize that suppressing head outputs is useful when feature groups carry little information at a given operating point, preventing uninformative features from accumulating through successive blocks.

Feature transforms. We replace SWA in the feature transforms with a linear projection, a 3×3 grouped convolution, and a second linear projection. This mirrors SWA's value projection, attention aggregation, and output projection, with 32 groups matching its attention heads. With the rest of our model unchanged, this replacement increases BD-rate by 1.71%, showing that the model benefits from content-adaptive spatial aggregation in the feature transforms.

Entropy distribution. Fixing $\beta = 2$ in the same GGD implementation increases BD-rate by 3.47%, a larger effect than the roughly 2% bitrate savings reported by Zhang et al. (2025) for recent compression architectures. Based on our latent analysis (Section 4.2.2), we hypothesize that SWA-IC benefits from uniform-like fits for inactive elements alongside Laplacian-to-Gaussian fits for rate-carrying elements.

Context features for synthesis. Removing context features from the synthesis input increases BD-rate by 8.53%, showing that the model benefits substantially from reusing context features for reconstruction. We hypothesize that these features provide a less noisy representation for the synthesis shortcut, reducing the propagation of quantization noise while preserving direct access to the decoded latent.

Channel partition. Replacing groups $(28, 56, 172)$ with nearly equal-sized groups $(85, 85, 86)$, retaining 256 channels and three channel-decoding stages per spatial step, increases BD-rate by 9.62%, the largest ablation effect. The growing groups carry comparable rates despite their unequal widths (Figure 4). We hypothesize that this partition better matches the channels' nonuniform information content by allocating fewer channels to the earlier, more information-dense groups.

5 Conclusion

We proposed SWA-IC, combining SWA-based feature transforms and context modeling with direct context-feature conditioning of synthesis and learned attention sinks. A single model with shared network weights across all rates supports 128 operating points and achieves state-of-the-art image compression. At medium and large tested resolutions, it also matches or improves on the decoding latency of previous state-of-the-art image codecs. Future research could investigate how rate adaptation can be improved at very low and very high bitrates.

AI USE STATEMENT

In this work, we used generative AI tools to draft and edit parts of the research paper to improve readability and clarity. We also used these tools to assist with retrieving information about related work. We additionally used an AI coding agent for code assistance with implementing and running baseline methods for benchmarking, and to help create and edit scientific figures and visualization scripts. For plots of experimental results, the authors supplied the locations of existing results and specified the quantities and effects to visualize; AI assistance was limited to implementing and editing the visualization scripts. The AI coding agent was not used to implement, modify, or otherwise develop our proposed method.

We did not use generative AI tools to analyze data, interpret results, or directly evaluate models. Model evaluation and scientific analysis were performed by the authors; AI involvement in benchmarking was limited to the code assistance described above. We also did not use generative AI tools for generating synthetic datasets, developing theoretical models, formulating mathematical claims, assisting in proofs, proposing hypotheses, designing experiments, cleaning datasets, or suggesting experimental parameters.

All AI-assisted text, code, and figures were reviewed and, where necessary, edited by the authors to ensure accuracy, correctness, originality, and alignment with the intended experimental setup. We take full responsibility for the content of this work, including all text, code, claims, results, and artifacts.

REPRODUCIBILITY STATEMENT

The appendix provides architecture specifications, training settings, and evaluation protocols, including details of entropy coding and runtime measurement. We plan to release our model implementation and trained checkpoints as open source upon acceptance to support reproduction of the reported results.

REFERENCES

Nicola Asuni and Andrea Giachetti. TESTIMAGES: A Large Data Archive For Display and Algorithm Testing. *Journal of Graphics Tools*, 17(4):113–125, July 2015. doi: 10.1080/2165347X.2015.1024298. URL https://www.tandfonline.com/doi/abs/10.1080/2165347X.2015.1024298.

Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end optimized image compression. In *International Conference on Learning Representations (ICLR)*, April 2017. URL https://openreview.net/forum?id=rJxdQ3jeg.

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 (ICLR)*, April 2018. URL https://openreview.net/forum?id=rkcQFMZRb.

Yunuo Chen, Bing He, Zezheng Lyu, Hongwei Hu, Qunshan Gu, Yuan Tian, and Guo Lu. Adaptive Learned Image Compression with Graph Neural Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12150–12161, June 2026a. URL https://openaccess.thecvf.com/content/CVPR2026/html/Chen_Adaptive_Learned_Image_Compression_with_Graph_Neural_Networks_CVPR_2026_paper.html.

Yunuo Chen, Zezheng Lyu, Bing He, Hongwei Hu, Qi Wang, Yuan Tian, Li Song, Wenjun Zhang, and Guo Lu. Content-Aware Mamba for Learned Image Compression. In *International Conference on Learning Representations (ICLR)*, April 2026b. URL https://openreview.net/forum?id=WwDNiisZQm.

Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Variable Rate Deep Image Compression With a Conditional Autoencoder. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3146–3154, October 2019. doi: 10.1109/ICCV.2019.00324. URL https://ieeexplore.ieee.org/document/9008820.

Ze Cui, Jing Wang, Shangyin Gao, Tiansheng Guo, Yihui Feng, and Bo Bai. Asymmetric Gained Deep Image Compression With Continuous Rate Adaptation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10527–10536, June 2021. doi: 10.1109/CVPR46437.2021.01039. URL https://openaccess.thecvf.com/content/CVPR2021/html/Cui_Asymmetric_Gained_Deep_Image_Compression_With_Continuous_Rate_Adaptation_CVPR_2021_paper.html.

Tri Dao. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *International Conference on Learning Representations (ICLR)*, May 2024. URL https://openreview.net/forum?id=mZn2Xyh9Ec.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255. IEEE, June 2009. URL https://ieeexplore.ieee.org/document/5206848.

Zhihao Duan, Ming Lu, Jack Ma, Yuning Huang, Zhan Ma, and Fengqing Zhu. QARV: Quantization-Aware ResNet VAE for Lossy Image Compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(1):436–450, January 2024. doi: 10.1109/TPAMI.2023.3322904. URL https://ieeexplore.ieee.org/document/10274142.

Donghui Feng, Zhengxue Cheng, Shen Wang, Ronghua Wu, Hongwei Hu, Guo Lu, and Li Song. Linear Attention Modeling for Learned Image Compression. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7623–7632, June 2025. doi: 10.1109/CVPR52734.2025.00714. URL https://openaccess.thecvf.com/content/CVPR2025/html/Feng_Linear_Attention_Modeling_for_Learned_Image_Compression_CVPR_2025_paper.html.

Rich Franzen. Kodak Lossless True Color Image Suite, 1999. URL http://r0k.us/graphics/kodak/.

Dailan He, Yaoyan Zheng, Baocheng Sun, Yan Wang, and Hongwei Qin. Checkerboard context model for efficient learned image compression. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14771–14780, June 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/He_Checkerboard_Context_Model_for_Efficient_Learned_Image_Compression_CVPR_2021_paper.html.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5718–5727, June 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/He_ELIC_Efficient_Learned_Image_Compression_With_Unevenly_Grouped_Space-Channel_Contextual_CVPR_2022_paper.html.

Christian Herglotz, Matthias Kränzler, Ruben Mons, and André Kaup. Beyond Bjøntegaard: Limits of Video Compression Performance Comparisons. In *IEEE International Conference on Image Processing (ICIP)*, pp. 46–50, October 2022. doi: 10.1109/ICIP46576.2022.9897912. URL https://ieeexplore.ieee.org/document/9897912.

Zhaoyang Jia, Bin Li, Jiahao Li, Wenxuan Xie, Linfeng Qi, Houqiang Li, and Yan Lu. Towards Practical Real-Time Neural Video Compression. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12543–12552, June 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Jia_Towards_Practical_Real-Time_Neural_Video_Compression_CVPR_2025_paper.html.

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, May 2025. doi: 10.1145/3719011. URL https://doi.org/10.1145/3719011.

JVET. Versatile Video Coding (VVC) Test Model 23.10 (VTM 23.10). https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/tree/VTM-23.10, May 2025.

Taesup Kim, Inchul Song, and Yoshua Bengio. Dynamic Layer Normalization for Adaptive Neural Acoustic Modeling in Speech Recognition. In *Interspeech*, pp. 2411–2415, August 2017. doi: 10.21437/Interspeech.2017-556. URL https://www.isca-archive.org/interspeech_2017/kim17f_interspeech.html.

Alexander Kopte and André Kaup. Sliding Window Attention for Learned Video Compression. In *Picture Coding Symposium (PCS)*, pp. 1–5, December 2025. doi: 10.1109/PCS65673.2025.11417665. URL https://ieeexplore.ieee.org/document/11417665.

Alexander Kopte and André Kaup. Parallel Context Modeling for Sliding Window Attention in Neural Video Coding. In *IEEE International Conference on Image Processing (ICIP)*, pp. 1–6, September 2026. doi: 10.1109/ICIP61757.2026.11630307. URL https://ieeexplore.ieee.org/document/11630307.

A. Burakhan Koyuncu, Han Gao, Atanas Boev, Georgii Gaikov, Elena Alshina, and Eckehard Steinbach. Contextformer: A transformer with spatio-channel attention for context modeling in learned image compression. In *European Conference on Computer Vision (ECCV)*, pp. 447–463, October 2022. doi: 10.1007/978-3-031-19800-7_26. URL https://www.ecva.net/papers/eccv_2022/papers_ECCV/papers/136790436.pdf.

Han Li, Shaohui Li, Wenrui Dai, Maida Cao, Nuowen Kan, Chenglin Li, Junni Zou, and Hongkai Xiong. On Disentangled Training for Nonlinear Transform in Learned Image Compression. In *International Conference on Learning Representations (ICLR)*, pp. 51785–51805, April 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/805c06617d2b643278936daadfde4280-Abstract-Conference.html.

Jiahao Li, Bin Li, and Yan Lu. Neural video compression with diverse contexts. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22616–22626, June 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Li_Neural_Video_Compression_With_Diverse_Contexts_CVPR_2023_paper.html.

Yuqi Li, Haotian Zhang, Li Li, and Dong Liu. Learned image compression with hierarchical progressive context modeling. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 18834–18843, October 2025b. URL https://openaccess.thecvf.com/content/ICCV2025/html/Li_Learned_Image_Compression_with_Hierarchical_Progressive_Context_Modeling_ICCV_2025_paper.html.

Jinming Liu, Heming Sun, and Jiro Katto. Learned Image Compression With Mixed Transformer-CNN Architectures. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14388–14397, June 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Liu_Learned_Image_Compression_With_Mixed_Transformer-CNN_Architectures_CVPR_2023_paper.html.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical vision transformer using shifted windows. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10012–10022, October 2021. URL https://openaccess.thecvf.com/content/ICCV2021/html/Liu_Swin_Transformer_Hierarchical_Vision_Transformer_Using_Shifted_Windows_ICCV_2021_paper.html.

Jingbo Lu, Leheng Zhang, Xingyu Zhou, Mu Li, Wen Li, and Shuhang Gu. Learned Image Compression with Dictionary-Based Entropy Model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12850–12859, June 2025. doi: 10.1109/CVPR52734.2025.01199. URL https://openaccess.thecvf.com/content/CVPR2025/html/Lu_Learned_Image_Compression_with_Dictionary-based_Entropy_Model_CVPR_2025_paper.html.

Fabian Mentzer, Eirikur Agustsson, and Michael Tschannen. M2T: Masking transformers twice for faster decoding. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5340–5349, October 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Mentzer_M2T_Masking_Transformers_Twice_for_Faster_Decoding_ICCV_2023_paper.html.

Evan Miller. Attention is off by one, July 2023. URL https://www.evanmiller.org/attention-is-off-by-one.html.

David Minnen and Saurabh Singh. Channel-wise autoregressive entropy models for learned image compression. In *IEEE International Conference on Image Processing (ICIP)*, pp. 3339–3343, October 2020. doi: 10.1109/ICIP40778.2020.9190935. URL https://ieeexplore.ieee.org/document/9190935.

David Minnen, Johannes Ballé, and George D. Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31. Curran Associates, Inc., December 2018. URL https://papers.nips.cc/paper_files/paper/2018/hash/53edebc543333dfbf7c5933af792c9c4-Abstract.html.

OpenAI. GPT-OSS-120B & GPT-OSS-20B model card, August 2025. URL https://cdn.openai.com/pdf/419b6906-9da6-406c-a19d-1bb078ac7637/oai_gpt-oss_model_card.pdf.

Alberto Presta, Enzo Tartaglione, Attilio Fiandrotti, and Marco Grangetto. STanH: Parametric Quantization for Variable Rate Learned Image Compression. *IEEE Transactions on Image Processing*, 34:639–651, January 2025. doi: 10.1109/TIP.2025.3527883. URL https://ieeexplore.ieee.org/document/10843163.

Yichen Qian, Xiuyu Sun, Ming Lin, Zhiyu Tan, and Rong Jin. Entroformer: A Transformer-based Entropy Model for Learned Image Compression. In *International Conference on Learning Representations (ICLR)*, April 2022. URL https://openreview.net/forum?id=VrjOFfcnSV8.

Myungseo Song, Jinyoung Choi, and Bohyung Han. Variable-Rate Deep Image Compression through Spatially-Adaptive Feature Transform. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2360–2369, October 2021. doi: 10.1109/ICCV48922.2021.00238. URL https://ieeexplore.ieee.org/document/9710010.

Calvin-Khang Ta, Praneet Singh, Tong Shao, and Peng Yin. MixCompress: Mixture of experts for variable rate learned image compression. In *European Conference on Computer Vision (ECCV)*, September 2026. URL https://eccv.ecva.net/virtual/2026/poster/3527.

Philippe Tillet, H. T. Kung, and David Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *ACM SIGPLAN International Workshop on Machine Learning and Programming Languages (MAPL)*, pp. 10–19, June 2019. URL https://dl.acm.org/doi/10.1145/3315508.3329973.

George Toderici, Wenzhe Shi, Radu Timofte, Lucas Theis, Johannes Ballé, Eirikur Agustsson, Nick Johnston, and Fabian Mentzer. Workshop and Challenge on Learned Image Compression (CLIC2020), June 2020. URL https://archive.compression.cc/challenge/.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open Foundation and Fine-Tuned Chat Models, July 2023. URL http://arxiv.org/abs/2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems (NIPS)*, volume 30. Curran Associates, Inc., December 2017. URL https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Fei Yang, Luis Herranz, Joost van de Weijer, José A. Iglesias Guitián, Antonio M. López, and Mikhail G. Mozerov. Variable Rate Deep Image Compression With Modulated Autoencoder. *IEEE Signal Processing Letters*, 27:331–335, January 2020. doi: 10.1109/LSP.2020.2970539. URL https://ieeexplore.ieee.org/document/8977394.

Biao Zhang and Rico Sennrich. Root Mean Square Layer Normalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32. Curran Associates, Inc., December 2019. URL https://papers.nips.cc/paper_files/paper/2019/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html.

Haotian Zhang, Li Li, and Dong Liu. Generalized Gaussian Model for Learned Image Compression. *IEEE Transactions on Image Processing*, 34:1950–1965, March 2025. doi: 10.1109/TIP.2025.3550013. URL https://ieeexplore.ieee.org/document/10938038.

Yinhao Zhu, Yang Yang, and Taco Cohen. Transformer-based transform coding. In *International Conference on Learning Representations (ICLR)*, April 2022. URL https://openreview.net/forum?id=IDwN6xjHnK8.

Renjie Zou, Chunfeng Song, and Zhaoxiang Zhang. The devil is in the details: Window-based attention for image compression. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17492–17501, June 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Zou_The_Devil_Is_in_the_Details_Window-Based_Attention_for_Image_CVPR_2022_paper.html.

A Appendix

A.1 Architecture and Training Details

Table 3 summarizes our architecture hyperparameters. The following paragraphs detail training, the transform and context blocks, and the decoding schedule.

Training implementation. We train with AdamW using $\epsilon = 10^{-8}$, otherwise default optimizer settings, and FP16/FP32 mixed precision. Each batch position has a fixed rate index; shuffling the images changes their rate assignments across epochs without separately permuting these indices. We drop the final incomplete batch so every update includes all rate points. Training takes approximately 13 hours and 40–45 minutes per epoch.

Data augmentation and reconstruction clipping. With equal probability, we either take a random 256×256 crop or take a random crop with scale $s \in [0.6, 1.0]$ and aspect ratio $r \in [0.75, 1.33]$ and resize it to 256×256. We also apply random horizontal flips. Reconstructions are not clipped during training; clipping to $[0, 1]$ is applied only for inference and evaluation.

Transform depth. Motivated by the decoder-heavy allocation of DCVC-RT (Jia et al., 2025), which uses 7 analysis and 13 synthesis blocks for its I-frame model and 3 analysis and 7 synthesis blocks for its P-frame model, we use 14 synthesis and seven analysis transformer blocks, giving a 2:1 ratio.

Transformer blocks. Following Touvron et al. (2023), we use pre-normalized residual SWA and SwiGLU sublayers with RMSNorm (Zhang & Sennrich, 2019). For an input $X \in \mathbb{R}^{N \times C}$ and $\bar{X} = \mathrm{RMSNorm}(X)$, a block computes

$$X' = X + \mathrm{SWA}(\bar{X}), \tag{9}$$

$$\bar{X}' = \mathrm{RMSNorm}(X'), \tag{10}$$

$$X_{\mathrm{out}} = X' + \left[\mathrm{SiLU}(\bar{X}'W_g) \odot (\bar{X}'W_u)\right] W_2, \tag{11}$$

where $W_g, W_u \in \mathbb{R}^{C\times mC}$, $W_2 \in \mathbb{R}^{mC\times C}$, and $m = 4$.

We use no dropout, stochastic depth, layer scaling, or positional encoding beyond the learned relative-position biases of SWA. Internal linear projections are bias-free, except for embedding layers, the terminal projections of the analysis and synthesis transforms, the final linear layer of each entropy-parameter head, and the wavelet-shortcut projections.

Wavelet shortcuts. The analysis shortcut bypasses the complete transformer branch, including its final 2×2 downsampling. A 2×2 Haar decomposition maps the patchified input $\boldsymbol{x}_f \in \mathbb{R}^{192\times \frac{H}{8}\times \frac{W}{8}}$ to four subbands with 768 channels in total. Following Li et al. (2025a), four channel-wise parameter vectors $\boldsymbol{s}_k$, initialized to 1, 0.5, 0.5, and 0, modulate the subbands by $\exp(\boldsymbol{s}_k)$. An orthogonally regularized linear projection maps their concatenation to the 256-channel latent.

The synthesis shortcut has independently learned parameters. Its orthogonally regularized projection maps $\boldsymbol{H}_{\text{syn}}$ from 1280 to 768 channels, which are split into four 192-channel subbands. These are scaled by independently learned factors $\exp(-\boldsymbol{s}_k)$ and transformed by the inverse Haar transform. The resulting 192-channel output is added to the nonlinear synthesis output before rearranging the patches into an RGB image.

For the orthogonality penalty, each shortcut projection matrix is oriented with its larger dimension as the row dimension, so that $W^\top W$ is a square matrix whose size equals the smaller projection dimension.

Gain initialization and quantization. We initialize all six gain banks to one. The four banks for the primary latent and its entropy parameters are lower-bounded at zero using gradient rectification, with no further constraints. For both the primary latent and hyperlatent, encoder and decoder gains are independently learned and are not constrained to be inverses. During training, the context model and synthesis transform receive $\hat{\boldsymbol{y}}$ formed using rounding and decoder gains, with straight-through gradients through rounding. Hyper-synthesis similarly receives $\hat{\boldsymbol{z}}$ formed using encoder gains, rounding with straight-through gradients, and decoder gains, while the hyperlatent rate loss uses additive uniform noise.

Ablation implementation. All ablation variants use the P-SWA spatial mask and include the hyperlatent gains. The "No SWA in transforms" ablation replaces SWA with grouped-convolution blocks only in the analysis and synthesis transforms; attention in the context model remains unchanged. The "No attention sinks" ablation removes sinks from all attention layers, including the context model.

GGD parameter constraints. Suppressing the spatial index for readability, we map the location and scale predictions to the quantized domain using independently learned gains:

$$\mu_{q,c} = g^{\mu}_{r,c}\mu_c, \qquad \alpha_{q,c} = \text{LowerBound}\left(g^{\alpha}_{r,c}\alpha_c, 0.125\right). \tag{12}$$

The shape parameter β_c is not rate-scaled.

We use the scale convention of Zhang et al. (2025); at $\beta = 2$, the corresponding Gaussian standard deviation is $\sigma = \alpha/\sqrt{2}$. The location parameter μ_c is unconstrained. Only the final scaled value $\alpha_{q,c}$ is lower-bounded; no output activation or earlier positivity constraint is applied to α_c. Unlike Zhang et al. (2025), we use a fixed scale lower bound of 0.125, independent of β. At the start of training, β_c is fixed to 2, corresponding to a Gaussian distribution. During the first $400\,000$ iterations, its admissible interval expands linearly from $[2, 2]$ to $[1, 7]$, after which it remains fixed at $[1, 7]$. We enforce the bounds on the gains, $\alpha_{q,c}$, and β_c through thresholding with gradient rectification: gradients that would move an out-of-range prediction farther from the admissible interval are suppressed, whereas gradients pointing back toward the interval are preserved.

Spatial context. We adapt the parallel context model of P-SWA (Kopte & Kaup, 2026) to image compression by removing its temporal context extractor and temporal cross-attention sublayers. A linear embedding followed by RMSNorm maps the decoder-scaled latent $\hat{\boldsymbol{y}}$ to the model dimension. Spatial Module 1 then applies N_{S_1} non-strictly masked transformer blocks and a final RMSNorm to produce S_1.

Hyperprior. The hyperprior maps S_1 to hyperlatent $\boldsymbol{z}$ using two downsampling residual blocks from DCVC-DC (Li et al., 2023), reducing each spatial dimension by a further factor of four. It is the only convolutional component, accounting for 0.63% of our model's parameters. Independent channel-wise encoder and decoder gains for each rate point produce

$$\tilde{\boldsymbol{z}} = \mathrm{round}(\boldsymbol{g}_{r,z}^{\mathrm{enc}} \odot \boldsymbol{z}), \qquad \hat{\boldsymbol{z}} = \boldsymbol{g}_{r,z}^{\mathrm{dec}} \odot \tilde{\boldsymbol{z}}. \tag{13}$$

The quantized hyperlatent $\tilde{\boldsymbol{z}}$ is transmitted before $\tilde{\boldsymbol{y}}$ using a learned factorized entropy model (Ballé et al., 2018), with a separate model for each rate point as in DCVC-RT (Jia et al., 2025). Two matching upsampling blocks, a linear embedding, and RMSNorm map $\hat{\boldsymbol{z}}$ to the hyperprior representation Z.

The hyperprior uses 64 channels and retains the biased convolutions of the DCVC residual blocks. Including its factorized entropy models, it contains $1.0\,\mathrm{M}$ of our model's $165.1\,\mathrm{M}$ parameters. Each of the 128 rate points has independent factorized distribution parameters and entropy-coding tables. The hyper-synthesis embedding and RMSNorm match its output to the spatial resolution and model dimension required by the accumulator.

Accumulator. The P-SWA accumulator combines Z with spatial context from S_1. It uses the same residual block structure, replacing self-attention with strictly masked cross-attention. Its residual stream starts from Z, with projections

$$Q^{(i)} = \bar{Z} W_Q^{(i)}, \quad K^{(i)} = S_1 W_K^{(i)}, \quad V^{(i)} = S_1 W_V^{(i)}, \tag{14}$$

where $\bar{Z} = \mathrm{RMSNorm}(Z)$ and S_1 is already normalized. Keys and values are restricted to previously decoded spatial steps. The accumulator consists of one residual cross-attention and SwiGLU block. Its output enters Spatial Module 2 without an additional embedding or normalization layer and passes through N_{S_2} non-strictly masked self-attention blocks, followed by RMSNorm and a linear projection, before entering the channel module. The accumulator is not counted toward N_{S_2}. Section 3.3 details the masking and decoding schedule.

Channel-group specification. Unlike P-SWA, we use growing channel groups with widths $(28, 56, 172)$, inspired by ELIC's uneven grouping (He et al., 2022) but using a different partition. Our choice of three groups was guided by preliminary observations that later groups predominantly represented high-frequency features, which we expected to offer less exploitable correlation. A larger final group therefore seemed a reasonable way to limit sequential decoding stages.

In general, we partition latent y into S groups:

$$\boldsymbol{y} = \mathrm{Concat}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_S), \qquad \boldsymbol{y}_i \in \mathbb{R}^{C_i \times \frac{H}{16} \times \frac{W}{16}}, \qquad \sum_{i=1}^{S} C_i = C. \tag{15}$$

Each group-specific branch embeds the output of Spatial Module 2 concatenated with the previously decoded groups $\hat{\boldsymbol{y}}_{<i} = \mathrm{Concat}(\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_{i-1})$, followed by RMSNorm. The first group receives only spatial context.

Following P-SWA, these branches use pre-normalized residual blocks with linear channel mixing in place of SWA. For group i, the mixing sublayer concatenates the normalized same-layer hidden states of groups $1, \ldots, i$ and projects them to width $H_i = eC_i$, followed by the residual update and SwiGLU sublayer. This is causal because each input embedding excludes its own undecoded symbols and all future groups. A final RMSNorm in each branch produces $\boldsymbol{H}_i$; together, these form

$$\boldsymbol{H}_{\mathrm{ctx}} = \mathrm{Concat}(\boldsymbol{H}_1, \ldots, \boldsymbol{H}_S), \qquad H_{\mathrm{ctx}} = \sum_{i=1}^{S} H_i = eC = 1024, \tag{16}$$

with $e = 4$. Three separate heads per group predict the GGD location, scale, and shape parameters directly from $\boldsymbol{H}_i$.

In the channel module, embeddings, mixing projections, and blocks are group-specific. Each mixing projection maps cumulative width $\sum_{j=1}^{i} H_j$ to H_i, preserving total width $\sum_i H_i$ across the branches. Each of the $3S$ entropy-parameter heads contains three linear layers separated by SiLU activations, with no hidden expansion or residual connections.

The "No growing channel groups" ablation retains the same hidden-width rule $H_i = eC_i$. Its parameter count consequently differs from our full model; the comparison evaluates the partition together with its associated allocation of prediction capacity.

Masking and decoding schedule. We retain the four-step diagonal wavefront pattern of P-SWA (Kopte & Kaup, 2026), shown in Figure 2.

Consider the output of Spatial Module 2 at position n. Its non-strict masking allows information from accumulator outputs at positions m satisfying $\tau(m) \leq \tau(n)$, where τ denotes the spatial prediction step. For each such position m, the accumulator's strict mask restricts keys and values to Spatial Module 1 outputs at positions p satisfying $\tau(p) < \tau(m)$. The non-strict masking within Spatial Module 1, in turn, restricts their dependence to input positions q satisfying $\tau(q) \leq \tau(p)$. Consequently, every input position contributing through this spatial context path satisfies

$$\tau(q) \leq \tau(p) < \tau(m) \leq \tau(n), \tag{17}$$

ensuring that only symbols from previously decoded spatial steps contribute to the output at n.

The hyperprior provides a separate information path that may depend on current and future positions. This remains valid because $\tilde{\boldsymbol{z}}$ is transmitted and decoded first.

Decoding follows the nested spatial–channel schedule of P-SWA. At each spatial step t, the context model uses all previously decoded spatial steps, and the channel module decodes the three groups of widths $(28, 56, 172)$ sequentially. Within each group, all positions assigned to t are decoded in parallel across the repeated tiles. Each decoded group becomes available to the next group at that step; after all three groups are decoded, processing advances to step $t + 1$. Decoding latent y therefore requires $4 \times 3 = 12$ sequential stages, preceded by hyperlatent decoding.

Latent residual prediction and stability. In the tested LRP configuration, the head was trained through the overall rate–distortion loss, without a separate loss for predicting quantization error or a constraint on its output range. We hypothesize that it could learn part of the synthesis mapping outside the stabilizing wavelet shortcut, contributing to the observed training instability.

Table 3: Architecture hyperparameters of SWA-IC.

Component	Hyperparameter	Value
Feature transforms	Encoder blocks (N_E)	7
	Decoder blocks (N_D)	14
	Attention heads (h)	32
	Head dimensions (d_k, d_v)	16
	Model dimension (d_{model})	512
	MLP expansion factor (m)	4
	Window size	3×3
Context model	Channel groups of latent y ($[C_1, C_2, \ldots, C_S]$)	$28, 56, 172$
	Spatial Module 1 blocks (N_{S_1})	4
	Spatial Module 2 blocks (N_{S_2})	4
	Channel-module blocks (N_C)	4
	Attention heads (h)	16
	Head dimensions (d_k, d_v)	32
	Spatial-module dimension (d_{model})	512
	Hyperprior dimension (M)	64
	Channel expansion factor (e)	4
	MLP expansion factor (m)	4
	Window size	11×11

A.2 Rate Estimation and Training Objective

Rate relaxation. For the rate loss, we replace rounding by additive uniform noise and use

$$\tilde{\boldsymbol{y}}^{\text{noise}} = \boldsymbol{g}_r^{\text{enc}} \odot \boldsymbol{y} + \boldsymbol{u}, \qquad u_c \sim \mathcal{U}\left(-\tfrac{1}{2}, \tfrac{1}{2}\right). \tag{18}$$

Let $F_{\mathrm{GGD}}(\cdot;\mu,\alpha,\beta)$ denote the generalized-Gaussian cumulative distribution function and let $\mathcal{C}_c$ denote the available spatial and channel context together with the decoded hyperprior. Suppressing the spatial index, the convolved density used during training is

$$p(\tilde{y}_c^{\mathrm{noise}} \mid \mathcal{C}_c) = F_{\mathrm{GGD}}\left(\tilde{y}_c^{\mathrm{noise}} + \tfrac{1}{2}; \mu_{q,c}, \alpha_{q,c}, \beta_c\right) - F_{\mathrm{GGD}}\left(\tilde{y}_c^{\mathrm{noise}} - \tfrac{1}{2}; \mu_{q,c}, \alpha_{q,c}, \beta_c\right). \tag{19}$$

Normalizing by the source-image area gives the rate of latent y in bits per pixel:

$$R(\tilde{\boldsymbol{y}}^{\mathrm{noise}}) = -\frac{1}{HW} \sum_{c,p} \log_2 p(\tilde{y}_{c,p}^{\mathrm{noise}} \mid \mathcal{C}_{c,p}), \tag{20}$$

where p indexes spatial positions. At inference, the CDF difference is evaluated at the quantized value $\tilde{y}_{c,p}$ for entropy coding.

We likewise add uniform noise to the encoder-scaled hyperlatent, yielding $\tilde{\boldsymbol{z}}^{\mathrm{noise}} = \boldsymbol{g}_{r,z}^{\mathrm{enc}} \odot \boldsymbol{z} + \boldsymbol{u}_z$, with each element of $\boldsymbol{u}_z$ drawn from $\mathcal{U}(-\frac{1}{2}, \frac{1}{2})$, and evaluate its rate under the corresponding factorized prior in bits per source-image pixel.

Distortion and rate–distortion objective. We optimize a rate–distortion objective using mean squared error (MSE) in RGB space as the distortion measure. For target images normalized to $[0, 1]$, we define

$$D(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{3HW} \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2. \tag{21}$$

For a batch of size b, with rate point r_n assigned to image n, the average rate–distortion (RD) objective is

$$L_{\mathrm{RD}} = \frac{1}{b} \sum_{n=1}^{b} \left[R(\tilde{\boldsymbol{y}}_n^{\mathrm{noise}}) + R(\tilde{\boldsymbol{z}}_n^{\mathrm{noise}}) + \lambda_{r_n} D(\boldsymbol{x}_n, \hat{\boldsymbol{x}}_n)\right]. \tag{22}$$

Rate weighting. We geometrically space the $P = 128$ Lagrangian multipliers between $\lambda_{\min} = 100$ and $\lambda_{\max} = 5000$:

$$\lambda_r = \lambda_{\min} \left(\frac{\lambda_{\max}}{\lambda_{\min}}\right)^{\frac{r}{P-1}}, \qquad r \in \{0, \ldots, P-1\}. \tag{23}$$

Each batch contains one sample per rate point ($b = P = 128$), so all operating points contribute to every update.

Shortcut regularization. Following Li et al. (2025a), we regularize the independently learned analysis and synthesis shortcut projections using

$$L_{\mathrm{orth}} = \sum_{W \in \mathcal{W}} \left\|W^\top W - I\right\|_F^2, \qquad \lambda_{\mathrm{orth}} = 0.1, \tag{24}$$

where $\mathcal{W}$ contains the analysis and synthesis shortcut projections. The complete objective is

$$L = L_{\mathrm{RD}} + \lambda_{\mathrm{orth}} L_{\mathrm{orth}}. \tag{25}$$

A.3 Additional Latent Analysis

Figure 6 shows the absolute and relative rate contributions on CLIC. The estimated total closely follows the actual bitstream rate. Figure 7 adds the per-channel heatmap, showing that increasing bitrate activates more channels, particularly in the largest group y_3.

Figure 8 separates the shape distributions by channel group. For each operating point and group, histograms pool elements across spatial positions and all CLIC images, using equal symbol weights or estimated code lengths, and are normalized separately. As more elements become active with increasing bitrate, the unweighted medians decrease, with the smallest reduction in y_3.

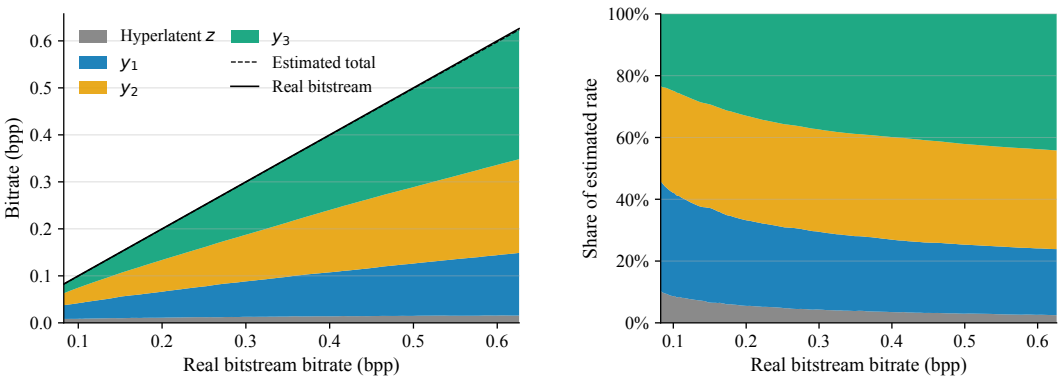

Figure 6: Rate allocation across operating points on CLIC: absolute rates (left) and relative shares (right). Black curves compare estimated and actual total rates.

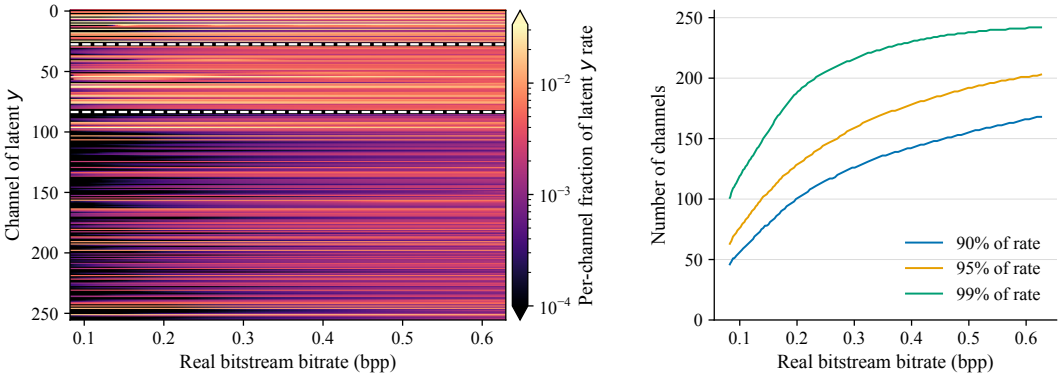

Figure 7: Channel-wise rate allocation on CLIC. Left: each channel's share of latent y rate, with dashed group boundaries. Right: highest-rate channels required to account for 90%, 95%, and 99% of latent y rate.

Comparison with prior GGD analysis. Zhang et al. (2025) restrict β to $[0.5, 4]$, whereas SWA-IC permits values up to 7 and develops a pronounced high-β population. Their shape analysis includes both all elements and a subset selected by estimated code length. Our "No GGD" ablation does not separately isolate the contributions of uniform-like fits for inactive elements and Laplacian-to-Gaussian fits for rate-carrying elements.

Scale constraints in the "No GGD" ablation. The "No GGD" ablation uses the same GGD implementation and scale lower bound $\alpha_{\min} = 0.125$ as our full model, with β fixed to 2 throughout training and evaluation. At $\beta = 2$, the corresponding standard deviation is $\sigma = \alpha/\sqrt{2}$. The comparison therefore retains the scale parameterization and its constraint while removing shape adaptation. At other values of β, a fixed α corresponds to a different standard deviation; the ablation measures the combined effects of shape adaptation on distribution tails and effective width.

A.4 Evaluation Protocol Details

VTM configuration. For VTM 23.10 (JVET, 2025), we use the `encoder_intra_vtm` All-Intra configuration with 8-bit YUV 4:4:4 input. FFmpeg conversion uses `-frames:v 1 -pix_fmt yuv444p -f rawvideo` to generate the VTM input and `-frames:v 1 -pix_fmt rgb24 -f rawvideo` for RGB evaluation. We test QPs $24, 26, \ldots, 50$ on Kodak, Tecnick, and CLIC.

BD-rate and local-rate comparison. We compute BD-rate using the conventional formulation, interpolating logarithmic bitrate as a function of PSNR and integrating over the overlapping quality interval. Following Herglotz et al. (2022), we use Akima interpolation instead of PCHIP. For the

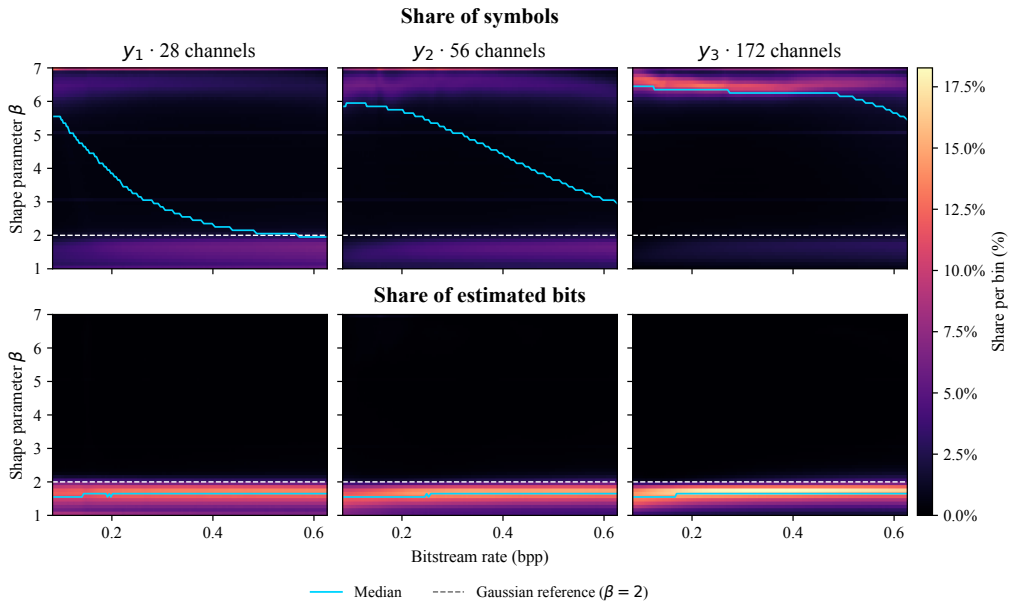

Figure 8: GGD shape distributions by channel group on CLIC. Color shows the share of symbols (top) or estimated bits (bottom) per β bin, using a shared scale. Each vertical distribution sums to 100%. Dashed lines mark $\beta = 2$; cyan curves show medians.

primary results, this interval is determined independently for each baseline and VTM pair. The local-rate curves use the same interpolation to compare SWA-IC, DCVC-RT-Intra, MixCompress-MoD, and LALIC with VTM at matched PSNR. Each curve spans that model's full measured quality overlap with VTM, without extrapolation.

Bitstreams and rate accounting. We generate actual bitstreams for all 128 operating points using our rANS implementation with 16-bit quantized CDFs. Reported bpp includes the bitstreams for latent y and hyperlatent z and is normalized by the original image area. Following the baseline evaluation protocols, the rate index and image dimensions are treated as externally supplied metadata and excluded from the reported rate.

Padding and dataset aggregation. We pad inputs to spatial dimensions divisible by 16 for the feature transforms. The hyperprior applies additional padding so that the latent spatial dimensions are divisible by four. We crop reconstructions to the original image dimensions before computing MSE and PSNR. At each operating point, PSNR and bpp are averaged separately across images; BD-rate is computed from these dataset-average curves.

Numerical precision. For codecs measured under our timing protocol, we use their official implementations under PyTorch AMP FP16, except TCM's entropy-coding path, which retains FP32, and DCVC-RT-Intra, which runs in FP16. All methods complete successfully under this protocol. For the baselines in Table 4, AMP changes PSNR by at most $0.018\,\mathrm{dB}$ and bitrate by at most 0.25% relative to FP32 across the three runtime images. These checks validate the AMP timing protocol; those baselines' rate–distortion comparisons retain their published FP32 results.

DCVC-RT execution. For the main comparisons, we evaluate DCVC-RT-Intra (Jia et al., 2025) in FP16 with its optional CUDA inference extensions disabled. These extensions combine operator fusion, kernels specialized to fixed operator configurations, and reduced Python and dispatch overhead—optimizations that we do not apply across the compared models. Our Triton implementation instead provides a generic sliding window attention operator supporting different window sizes, masks, head configurations, and batch sizes, while retaining PyTorch dispatch and separate execution of surrounding layers. Table 6 additionally reports timings with the optional extensions enabled.

Table 4: Validation of the mixed-precision runtime protocol. Deltas are AMP minus FP32. TCM (∗) uses AMP for analysis and synthesis and FP32 for its entropy-coding path. For the daggered LALIC and DCAE entries, FP32 uses cuDNN disabled, whereas AMP uses cuDNN with benchmarking enabled.

Model	Resolution	FP32 PSNR	AMP PSNR	Δ PSNR	FP32 bpp	AMP bpp	Δ bpp (%)
MLIC++	768×512	34.224	34.231	0.007	1.055 420	1.055 827	0.04
MLIC++	1200×1200	36.863	36.853	−0.009	0.739 644	0.739 622	0.00
MLIC++	2048×2048	35.466	35.466	0.000	0.138 412	0.138 367	−0.03
TCM Large*	768×512	36.745	36.749	0.003	1.498 942	1.499 512	0.04
TCM Large*	1200×1200	39.998	39.980	−0.018	0.941 444	0.941 133	−0.03
TCM Large*	2048×2048	36.782	36.781	−0.001	0.360 657	0.360 817	0.04
HPCM Large	768×512	31.720	31.715	−0.005	0.668 213	0.668 294	0.01
HPCM Large	1200×1200	35.194	35.185	−0.008	0.298 156	0.297 867	−0.10
HPCM Large	2048×2048	34.774	34.775	0.001	0.068 207	0.068 153	−0.08
LALIC†	768×512	32.096	32.087	−0.009	0.751 790	0.750 488	−0.17
LALIC	1200×1200	35.227	35.235	0.007	0.334 444	0.334 844	0.12
LALIC	2048×2048	34.933	34.933	0.000	0.082 245	0.082 230	−0.02
DCAE†	768×512	32.163	32.160	−0.004	0.749 349	0.748 779	−0.08
DCAE	1200×1200	35.680	35.662	−0.018	0.382 356	0.381 956	−0.10
DCAE	2048×2048	34.957	34.955	−0.001	0.081 726	0.081 604	−0.15
GLIC	768×512	32.188	32.178	−0.010	0.739 095	0.738 525	−0.08
GLIC	1200×1200	35.286	35.280	−0.006	0.310 022	0.309 244	−0.25
GLIC	2048×2048	34.946	34.941	−0.005	0.082 184	0.081 993	−0.23
CMIC	768×512	32.146	32.145	−0.001	0.742 269	0.742 676	0.05
CMIC	1200×1200	35.283	35.287	0.004	0.338 378	0.337 756	−0.18
CMIC	2048×2048	34.938	34.936	−0.003	0.079 277	0.079 224	−0.07

Hardware and release configuration. All runtime benchmarks use an Intel Core i9-10900X CPU, with all CPU cores available to each method. We plan to include the timing harness, exact checkpoint identifiers, and detailed configurations in the code release.

Runtime inputs and checkpoints. Runtime is evaluated on the Kodak image `kodim01` (Franzen, 1999), the TESTIMAGES ART image `ART_OR_1200x1200_001` (Asuni & Giachetti, 2015), and the CLIC image `allef-vinicius-109434` (Toderici et al., 2020). The CLIC image is reflect-padded to 2048×2048. For the learned baselines, we use downloaded checkpoints.

Timing procedure. We use a unified timing harness. We transfer inputs to the GPU before timing. Method-specific padding and arithmetic coding are included, whereas image loading, file I/O, and host-to-device transfer are excluded. Each measurement consists of one warm-up execution followed by five explicitly synchronized executions; Table 5 reports their median. We repeated the measurements on a second machine with an identical configuration and obtained closely matching results.

cuDNN configuration. We set `cudnn.benchmark=True` for every method, allowing the convolutional baselines to select faster kernels. Disabling this option has no measurable runtime effect on our model. LALIC and DCAE produce invalid FP32 reconstructions on the Kodak runtime image with cuDNN enabled; disabling cuDNN resolves the issue. Their corresponding AMP executions use cuDNN with benchmarking enabled and produce valid reconstructions.

A.5 Model Size, Runtime, and Computational Complexity

Runtime and model size. Table 5 reports the complexity comparison. SWA-IC encodes faster than the evaluated single-rate codecs at 1200×1200 and 2048×2048; DCVC-RT-Intra is faster at all three resolutions. Following DCVC-RT (Jia et al., 2025), SWA-IC concentrates transform computation in wide representations at low spatial resolution.

Table 5: Model size, neural-operation counts, and encoding/decoding latency on an RTX 3090. Timings include arithmetic coding; MAC counts exclude it. Dashes indicate unreported counts.

Method	Params/model (M)	kMACs/px		Runtime (s) ↓					
		Enc.	Dec.	768×512		1200×1200		2048×2048	
				Enc.	Dec.	Enc.	Dec.	Enc.	Dec.
Single-rate models									
TCM Large (Liu et al., 2023)	76.6	813.5	1153.3	0.139	0.163	0.419	0.453	0.970	1.060
MLIC++ (Jiang et al., 2025)	83.5	577.3	777.8	0.143	0.202	0.301	0.385	0.736	0.801
DCAE (Lu et al., 2025)	119.4	645.3	642.6	0.096	0.115	0.304	0.341	0.838	0.851
GLIC (Chen et al., 2026a)	69.4	665.7	726.3	0.262	0.186	0.730	0.568	1.797	1.418
CMIC (Chen et al., 2026b)	69.8	586.6	583.7	0.240	0.164	0.550	0.399	1.430	1.056
HPCM Large (Li et al., 2025b)	89.7	1012.3	1006.8	0.059	0.058	0.189	0.176	0.488	0.412
LALIC (Feng et al., 2025)	66.1	385.2	377.8	0.259	0.181	0.675	0.507	1.630	1.246
Multirate models									
DCVC-RT-Intra (Jia et al., 2025)	**45.7**	**258.4**	**362.2**	**0.014**	**0.021**	**0.041**	**0.057**	**0.102**	**0.141**
SWA-IC (Ours)	165.1	748.5	1219.6	0.096	0.106	0.138	0.176	0.251	0.362

Table 6: DCVC-RT-Intra with optional CUDA inference extensions enabled on an RTX 3090. Speedups are relative to the PyTorch timings in Table 5.

Resolution	Time (ms)		Speedup (×)	
	Enc.	Dec.	Enc.	Dec.
768 × 512	9.65	13.71	1.40	1.52
1200 × 1200	25.89	34.87	1.58	1.62
2048 × 2048	60.49	79.69	1.68	1.77

Although SWA-IC has more parameters per checkpoint than the evaluated single-rate codecs, it supports all 128 operating points within one checkpoint containing 165.1 M parameters. Most evaluated single-rate baselines use six separately trained checkpoints, multiplying their per-model parameter counts by six to cover their reported operating points.

DCVC-RT with CUDA inference extensions. Enabling the optional CUDA inference extensions reduces DCVC-RT-Intra's encoding time by factors of 1.40–1.68 and decoding time by factors of 1.52–1.77 under the same timing protocol (Table 6). Evaluation at 1200×1200 required an alignment fix in two kernels, selecting the existing scalar CUDA path for unaligned tensor slices.

Operation counting. Parameter counts include trainable parameters only. Neural-operation counts are evaluated on a 2048×2048 input and exclude arithmetic coding. We supplement PyTorch profiler counts with explicit complexity estimates for custom operators, implementing the formulas reported in the respective papers. For the query–key and attention–value products of SWA, we count

$$\mathrm{MACs}_{\mathrm{SWA}} = bh(1 + 2H_w)(1 + 2W_w)s_q(d_k + d_v), \tag{26}$$

where b is the batch size, h is the number of heads, s_q is the number of queries per head and image, and d_k and d_v are the query/key and value dimensions, respectively. The vertical and horizontal window radii are H_w and W_w, giving attention-window dimensions $(1+2H_w)$ and $(1+2W_w)$. We count two FLOPs per multiply–accumulate operation.

A.6 Additional Rate–Distortion Results

Figure 9 provides the rate–distortion curves and local bitrate differences for Kodak, Tecnick, and CLIC, extending the CLIC comparison in Figure 3.

Table 7 compares the codecs over the fixed common quality intervals specified below and over their middle thirds. SWA-IC ranks first under both measures, but its BD-rate gap to the strongest learned baseline is larger in the middle third on every dataset. Its savings relative to VTM also increase over this interval. These results reinforce the trend in Figure 9: SWA-IC's relative advantage is strongest in the middle range, with weaker performance toward the ends, particularly at high rates.

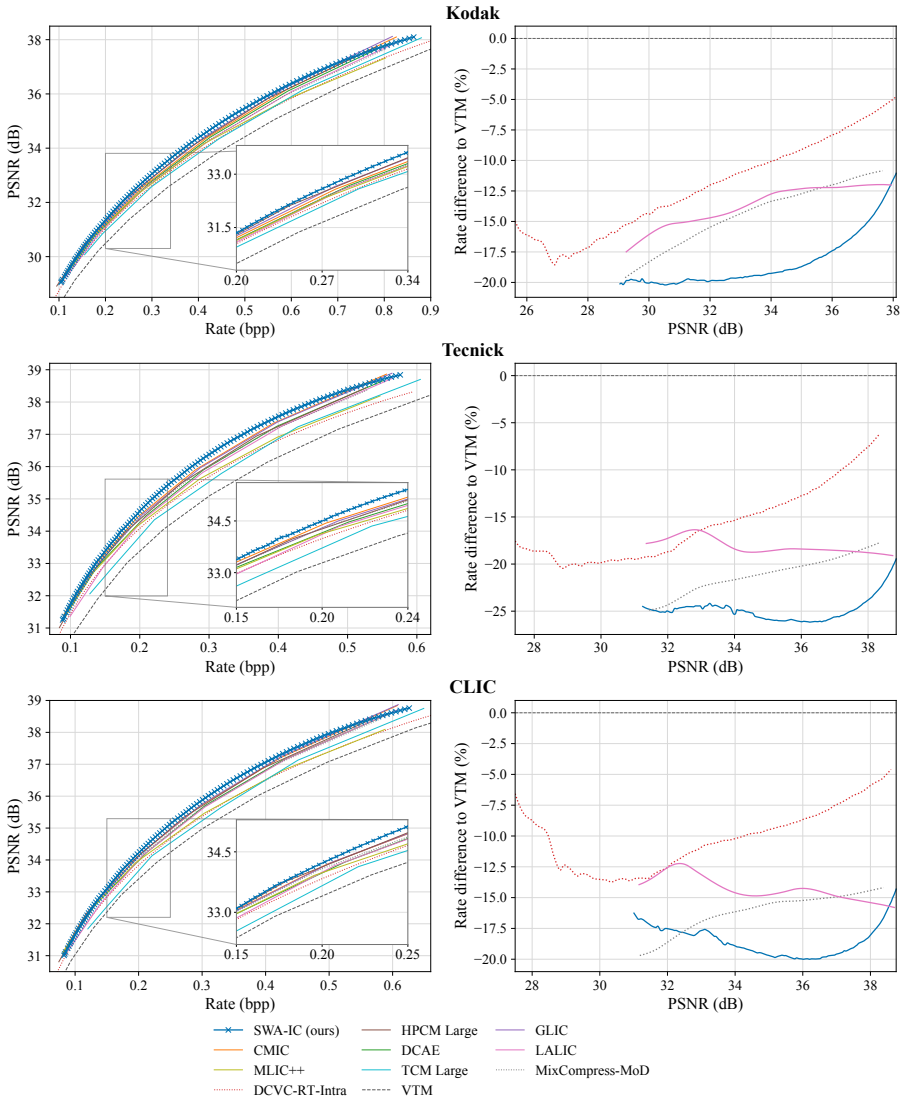

Figure 9: Rate–distortion curves for Kodak, Tecnick, and CLIC (left; dotted lines denote multirate baselines) and local bitrate differences relative to VTM 23.10 (right). Local differences span each model's full measured quality overlap with VTM.

DCVC-RT-Intra achieves larger savings relative to VTM over its full pairwise interval than over the common interval on Kodak and Tecnick; on CLIC, its savings are larger over the common interval. Since the full-range comparisons against VTM can use different quality intervals, their reported BD-rates should not be subtracted to estimate the saving between codecs. Direct pairwise comparisons against DCVC-RT-Intra yield SWA-IC savings of 10.03%, 12.85%, and 10.02% on Kodak, Tecnick, and CLIC, respectively.

Table 7: BD-rate (%) relative to VTM 23.10 (JVET, 2025) over the quality interval common to the methods in this table and over its middle third; lower is better. The common PSNR intervals are $[30.07, 37.32]$ dB for Kodak, $[32.06, 38.19]$ dB for Tecnick, and $[31.84, 38.09]$ dB for CLIC.

	Full common interval			Middle third		
Method	**Kodak**	**Tecnick**	**CLIC**	**Kodak**	**Tecnick**	**CLIC**
Single-rate models						
TCM Large (Liu et al., 2023)	−10.28	−11.30	−9.04	−9.83	−11.11	−8.92
MLIC++ (Jiang et al., 2025)	−12.58	−16.76	−13.50	−13.13	−16.65	−13.01
DCAE (Lu et al., 2025)	−15.02	−20.25	−16.69	−14.47	−19.40	−16.00
GLIC (Chen et al., 2026a)	−17.57	−22.22	−18.80	−17.27	−21.99	−18.55
CMIC (Chen et al., 2026b)	−16.12	−23.33	−18.82	−15.63	−23.05	−18.48
HPCM Large (Li et al., 2025b)	−17.86	−21.51	−19.06	−17.46	−21.33	−18.80
LALIC (Feng et al., 2025)	−13.48	−18.08	−15.45	−13.20	−18.55	−15.89
Multirate models						
DCVC-RT-Intra (Jia et al., 2025)	−10.37	−13.56	−10.68	−10.35	−13.99	−10.82
MixCompress-MoD (Ta et al., 2026)	−14.06	−20.87	−17.13	−13.73	−20.84	−16.83
SWA-IC (Ours)	**−18.80**	**−25.19**	**−20.13**	**−19.34**	**−25.71**	**−20.72**