Bound-Conditioned Latent Inference for Progressive Image Compression

Jaeseok Jang[†]
Korea University
jsjang@mcl.korea.ac.kr

Seungmin Jeon[†]
Korea University
seungminjeon@mcl.korea.ac.kr

Kwang Pyo Choi
Samsung Electronics
kp5.choi@samsung.com

Chang-Su Kim[*]
Korea University
changsukim@korea.ac.kr

Abstract

Progressive image compression reconstructs images from prefixes of a single bitstream and therefore involves sequential latent inference under partial observations. Existing trit-plane codecs mainly treat decoded trits as discrete symbols for probability prediction, without fully exploiting their role as interval constraints on the underlying continuous latents. We propose bound-conditioned latent inference (BLI), a framework that reformulates progressive trit-plane coding as sequential latent inference under shrinking interval constraints. At each coding step, the trits decoded so far define lower and upper bounds for each latent element. BLI uses the center and width of these bounds, together with hyperprior information shared by the encoder and decoder, to jointly refine the Gaussian parameters (μ, σ) for entropy modeling and the latent estimate $\hat{y}$ for reconstruction. As additional trits are decoded, the feasible intervals shrink monotonically, yielding tighter constraints for the next inference step and enabling flexible in-plane refinement schedules. Experiments on Kodak, CLIC, and JPEG-AI show that BLI achieves state-of-the-art rate-distortion performance among progressive image codecs, reducing BD-rate by 17.78% over DPICT and by 7.96% over CTC on Kodak, while replacing CTC's plane-specific predictors with a shared model that uses about $5.3\times$ fewer parameters and achieves lower latency than CTC.

1 Introduction

Progressive image compression differs from standard fixed-rate compression in that it allows the decoder to reconstruct an image from prefixes of a single bitstream. At an intermediate decoding step, only part of the latent representation has been transmitted, while the unresolved latent values must be inferred from partial observations. Thus, progressive compression is not merely a rate-distortion (R-D) optimization problem, but also a sequential latent inference problem under partial observations. This property is essential for adaptive streaming, bandwidth-limited transmission, and storage systems that require multiple quality levels from a single bitstream.

Traditional and learned codecs have achieved strong R-D performance, from hand-crafted transform codecs such as JPEG and JPEG2000 (Wallace, 1992; Skodras et al., 2001) to neural codecs with learned transforms, hyperpriors, and context models (Agustsson et al., 2017; Ballé et al., 2017, 2018; Minnen et al., 2018; He et al., 2021). However, most high-performance learned codecs are optimized

[†]Jaeseok Jang and Seungmin Jeon contributed equally to this work and are co-first authors.
[*]Corresponding author: Chang-Su Kim, <changsukim@korea.ac.kr>

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

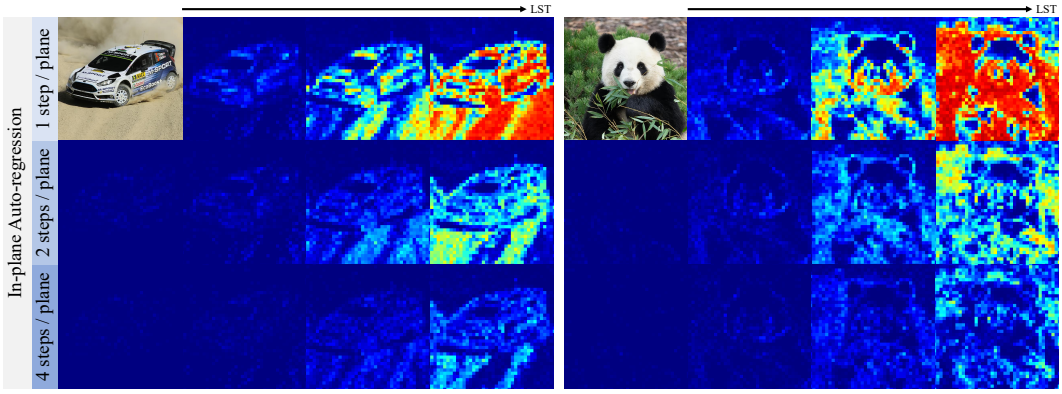


Figure 1: Rate-allocation maps of the proposed bound-conditioned latent inference (BLI) codec. Each column corresponds to the same trit-plane level, *i.e.*, the same degree of latent quantization, while each row uses a different number of in-plane refinement steps. The top row encodes an entire trit-plane in a single step, whereas the middle and bottom rows divide each trit-plane into two and four substeps, respectively. Colors indicate spatial rate allocation, with warmer colors corresponding to higher coding rates and cooler colors to lower rates. More frequent bound-conditioned updates iteratively refine the latent distribution within the same quantization level, yielding better rate efficiency without changing the final trit-plane precision.

for a single target rate. Multirate codecs (Cui et al., 2021; Yang et al., 2021; Xu et al., 2025) improve rate flexibility, but generally require separately encoded bitstreams for different rates, in contrast to progressive codecs, every bitstream prefix of which remains decodable.

Learned progressive codecs provide decodable bitstream prefixes by decomposing latent representations into successively refinable components. PLONQ (Lu et al., 2021) uses nested quantization grids for coarse-to-fine refinement, while DPICT (Lee et al., 2022) discretizes each latent element into trits that are transmitted sequentially. CTC (Jeon et al., 2023) further improves trit-plane coding by introducing context models that refine ternary probabilities and partial reconstructions. However, existing trit-plane codecs still primarily treat decoded trits as discrete symbols for probability prediction and reconstruction update. In particular, CTC improves discrete trit prediction and partial reconstruction, but it does not explicitly perform in-plane continuous latent inference conditioned on the feasible intervals induced by previously decoded trits. The key observation behind our approach is that decoded trits are not merely discrete symbols: in trit-plane coding, they also restrict each underlying continuous latent element to a feasible interval, which can serve as an explicit constraint for latent inference.

We present bound-conditioned latent inference (BLI), a framework that explicitly exploits these feasible intervals by reformulating progressive trit-plane coding as sequential latent inference under shrinking interval constraints. In BLI, each partially decoded latent element is modeled as a continuous variable constrained by lower and upper bounds determined by the trits decoded so far. At each coding step, BLI takes the center and width of the current feasible interval, together with hyperprior information shared by the encoder and decoder, and jointly refines the Gaussian parameters (μ, σ) for entropy modeling and the latent estimate $\hat{y}$ for reconstruction. As additional trits are decoded, the feasible intervals shrink monotonically, yielding tighter constraints for the next inference step. As illustrated in Figure 1, this bound-conditioned formulation naturally supports flexible in-plane transmission schedules: each trit-plane can be encoded at once or divided into finer substeps, allowing the model to update the latent distribution and latent estimate more frequently. Conceptually, BLI can be interpreted as constraint-consistent latent refinement, related to projection-based feasibility methods such as POCS (Bauschke and Borwein, 1996) and to denoising or inpainting under interval constraints. Unlike exact metric projection, however, BLI uses a learned amortized estimator trained directly for entropy modeling and reconstruction quality.

Our main contributions are summarized as follows:

- We reformulate progressive trit-plane coding as sequential latent inference under shrinking interval constraints, where decoded trits define feasible ranges for the underlying continuous latent elements.

- We propose BLI, a learned constrained inference framework that jointly refines entropy-model parameters and latent estimates from bound and hyperprior information, with flexible in-plane autoregressive updates within each trit-plane.
- Experiments show that BLI achieves state-of-the-art R-D performance among progressive image codecs, reducing BD-rate by 17.78% over DPICT and by 7.96% over CTC on the Kodak dataset, while replacing CTC's plane-specific prediction networks with a shared model that uses about $5.3\times$ fewer parameters and achieves lower latency than CTC.

2 Related Work

Learned image compression: Traditional codecs (Wallace, 1992; Bellard, 2014; Bross et al., 2021) follow the transform coding paradigm (Goyal, 2001), while learned codecs replace hand-crafted transforms and entropy models with neural networks. Early learned codecs introduced differentiable quantization and end-to-end R-D optimization (Agustsson et al., 2017; Ballé et al., 2017), and hyperprior-based models improved entropy estimation by transmitting side information about latent distributions (Ballé et al., 2018). Later methods further developed expressive priors, including Gaussian mixture and asymmetric Gaussian models (Cheng et al., 2020; Cui et al., 2021).

Entropy models improve probability estimation by exploiting dependencies among latent elements. Autoregressive context models (Minnen et al., 2018; Mentzer et al., 2018; Lee et al., 2019) improve compression efficiency through sequential prediction, but incur high coding latency due to limited parallelism. Partially parallel models (Minnen and Singh, 2020; He et al., 2021) reduce this cost by exploiting latent correlations more efficiently.

Recent learned codecs incorporate attention mechanisms and transformers into analysis-synthesis networks and entropy models. Transformer-based codecs (Zhu et al., 2022; Zou et al., 2022; Liu et al., 2023; Li et al., 2024) improve global context modeling and yield compact latents, while transformer-based entropy models (Qian et al., 2022; Kim et al., 2022) capture long-range dependencies for more accurate probability estimation. These advances provide strong fixed-rate baselines, but most of them are not designed to produce a single progressively decodable bitstream.

Scalable image compression: Beyond fixed-rate learned codecs, scalable compression aims to support multiple bitrates using a single trained model or a single embedded bitstream. Variable-rate methods adjust the operating rate by modulating internal representations, such as gain parameters, channel width, or spatial transforms (Cui et al., 2021; Yang et al., 2021; Song et al., 2021), or by using adjustable quantizers (Chen and Ma, 2020; Tong et al., 2023; Xu et al., 2025). Although these methods provide flexible rate control from a single model, each target bitrate generally requires a separately encoded bitstream. Therefore, they differ from progressive codecs, in which every prefix of a single bitstream remains decodable at multiple quality levels.

Progressive image compression uses embedded bitstreams, where reconstruction quality improves as additional bits are appended. Classical codecs such as progressive JPEG (Wallace, 1992), JPEG2000 (Skodras et al., 2001), and SPIHT (Said and Pearlman, 1996) achieve this through spectral selection, embedded bitstreams, or significance-driven coefficient ordering. In learned compression, PLONQ (Lu et al., 2021) introduced nested quantization for coarse-to-fine refinement, while DPICT (Lee et al., 2022) realized fine-grained scalability by slicing latents into trit-planes. CTC (Jeon et al., 2023) further improved ternary probability modeling and partial reconstructions. However, DPICT and CTC still primarily model decoded trits as discrete symbols. As a result, the feasible intervals induced by decoded trits are not explicitly exploited for in-plane continuous latent inference, which is the key limitation addressed by BLI.

3 Proposed Algorithm

We present bound-conditioned latent inference (BLI) for progressive trit-plane coding. BLI treats each partially decoded latent element as a continuous variable constrained by lower and upper bounds determined by the trits decoded so far. Within these bounds, BLI jointly refines the Gaussian parameters (μ, σ) for entropy modeling and the latent estimate $\hat{y}$ for reconstruction as the bounds shrink over coding steps. Unlike CTC (Jeon et al., 2023), which improves ternary probability modeling and partial reconstruction through discrete context prediction, BLI performs bound-conditioned Gaussian

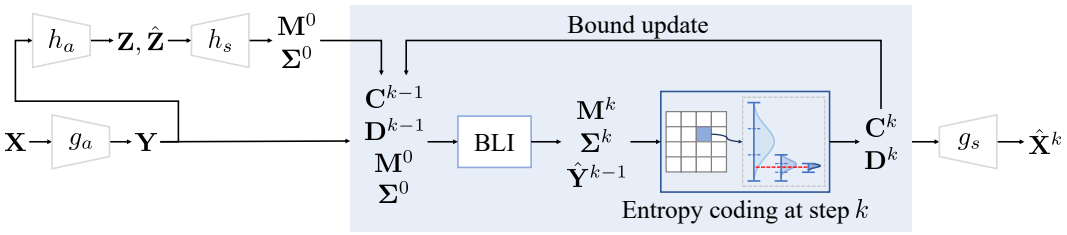


Figure 2: Overview of the BLI process. At coding step k, the trits decoded so far define bound tensors $(\mathbf{C}^{k-1}, \mathbf{D}^{k-1})$, which represent the current feasible ranges of latent elements. Given these bounds and the hyperprior parameters $(\mathbf{M}^0, \mathbf{\Sigma}^0)$, BLI estimates the Gaussian parameters $(\mathbf{M}^k, \mathbf{\Sigma}^k)$ for entropy coding and the latent tensor $\hat{\mathbf{Y}}^{k-1}$ for reconstruction under the current bounds. After a subset of trits is decoded using $(\mathbf{M}^k, \mathbf{\Sigma}^k)$, the bounds are updated to $(\mathbf{C}^k, \mathbf{D}^k)$, yielding tighter feasible ranges that, in turn, are used in the next coding step. This step-wise process iteratively refines the latent distribution and reconstruction.

inference at every refinement step. This enables iterative in-plane refinement of both the latent distribution and latent estimate, improving entropy modeling accuracy and reconstruction quality.

3.1 Bound-Conditioned Latent Inference

Suppose that an image $\mathbf{X}$ is transformed into a latent tensor $\mathbf{Y} \in \mathbb{R}^{C\times H\times W}$, as illustrated in Figure 2. Each element y_i in $\mathbf{Y}$ is represented by trits and progressively transmitted. BLI proceeds in coding steps indexed by a global counter $k = 1, 2, \cdots$, which increases monotonically across trit-planes. At step k, we define the following state variables.

- $(\mathbf{M}^k, \mathbf{\Sigma}^k)$: Gaussian mean and standard deviation parameters used for entropy coding at step k.
- $\hat{\mathbf{Y}}^k$: Reconstructed latent estimate consistent with the bounds after step k.
- $(\mathbf{L}^k, \mathbf{U}^k)$: Lower and upper bounds of the latent elements after step k.
- $(\mathbf{C}^k, \mathbf{D}^k)$: Centers and widths of the feasible intervals after step k, defined as

$$\mathbf{C}^k = \frac{\mathbf{L}^k + \mathbf{U}^k}{2}, \quad \mathbf{D}^k = \mathbf{U}^k - \mathbf{L}^k. \tag{1}$$

Note that a hyperprior entropy model initializes $(\mathbf{M}^0, \mathbf{\Sigma}^0)$, while $(\mathbf{L}^0, \mathbf{U}^0)$, and equivalently $(\mathbf{C}^0, \mathbf{D}^0)$, define the initial feasible support of the latent elements.

At step k, the current feasible interval of y_i is given by $\mathcal{I}_i^{k-1} = [\mathbf{L}_i^{k-1}, \mathbf{U}_i^{k-1}]$. This interval is equally divided into three sub-intervals $\mathcal{I}_{i,t}^{k-1}$ for $t = 0, 1, 2$, corresponding to the next undetermined trit at location i. BLI takes the current bound tensors $(\mathbf{C}^{k-1}, \mathbf{D}^{k-1})$ and the hyperprior parameters $(\mathbf{M}^0, \mathbf{\Sigma}^0)$ as inputs, and estimates the Gaussian parameters $(\mathbf{M}^k, \mathbf{\Sigma}^k)$ for entropy coding at step k. If y_i lies in the t_i-th sub-interval, the corresponding trit t_i is encoded by an arithmetic coder using

$$-\log_2 P(y_i \in \mathcal{I}_{i,t_i}^{k-1} \,|\, \mathbf{M}_i^k, \mathbf{\Sigma}_i^k) \tag{2}$$

bits. Accordingly, BLI is trained to estimate $(\mathbf{M}^k, \mathbf{\Sigma}^k)$ by minimizing the rate loss

$$\mathcal{L}_R^k = -\textstyle\sum_i \log_2 P(y_i \in \mathcal{I}_{i,t_i}^{k-1} \,|\, \mathbf{M}_i^k, \mathbf{\Sigma}_i^k). \tag{3}$$

In addition to the Gaussian parameters, BLI estimates a latent reconstruction within the current bounds by predicting a residual correction to the interval center:

$$\hat{\mathbf{Y}}^{k-1} = \mathbf{C}^{k-1} + \Delta\hat{\mathbf{Y}}^{k-1}. \tag{4}$$

Here, the indices reflect the timing of the coding step; at step k, the Gaussian parameters $(\mathbf{M}^k, \mathbf{\Sigma}^k)$ are used to encode or decode the next subset of trits, whereas $\hat{\mathbf{Y}}^{k-1}$ is the latent estimate consistent with the bounds $(\mathbf{C}^{k-1}, \mathbf{D}^{k-1})$ available prior to coding that subset. After the subset is coded and the bounds are updated, the corresponding $\hat{\mathbf{Y}}^k$ is estimated and used for reconstruction. The Gaussian

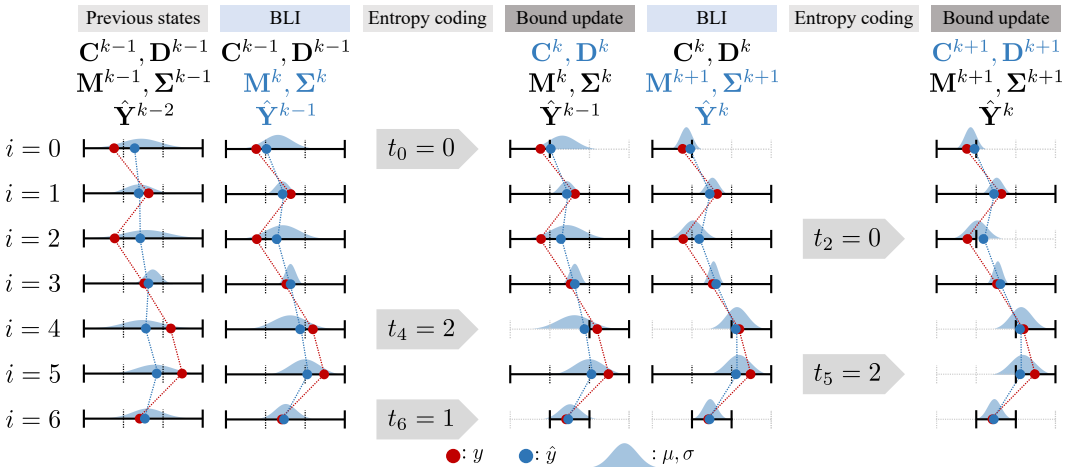


Figure 3: Illustration of the BLI refinement process using a 1D sequence of latent elements. Given the current bounds $(\mathbf{C}^{k-1}, \mathbf{D}^{k-1})$, BLI estimates $(\mathbf{M}^k, \mathbf{\Sigma}^k)$ for entropy coding and $\hat{\mathbf{Y}}^{k-1}$ for reconstruction under those bounds. After selected trits are decoded, the bounds are updated to $(\mathbf{C}^k, \mathbf{D}^k)$, yielding tighter feasible intervals for the next BLI step. Red dots denote original latent values, blue dots denote latent estimates, and blue densities denote estimated Gaussian distributions.

parameters and latent estimates are produced from shared attention features but through separate prediction heads. The distortion loss is then defined as

$$\mathcal{L}_D^k = \|\mathbf{X} - \hat{\mathbf{X}}^k\|_2^2 = \|\mathbf{X} - g_s(\hat{\mathbf{Y}}^k)\|_2^2. \tag{5}$$

The subset of trits is coded using $(\mathbf{M}^k, \mathbf{\Sigma}^k)$, and the decoded trits update the bounds from $(\mathbf{C}^{k-1}, \mathbf{D}^{k-1})$ to $(\mathbf{C}^k, \mathbf{D}^k)$.

Figure 3 illustrates the refinement process using a 1D sequence of latent elements. At step k, the decoded trits define interval constraints $\mathcal{I}_i^{k-1} = [\mathbf{L}_i^{k-1}, \mathbf{U}_i^{k-1}]$ for each latent element, and BLI performs learned constrained inference within these intervals. As additional trits are decoded, the intervals shrink monotonically, yielding tighter constraints for the next inference step. This process is conceptually related to denoising or inpainting under range constraints and to projection-based feasibility methods such as POCS (Bauschke and Borwein, 1996), since each step refines the estimates while enforcing consistency with known constraints. Unlike exact metric projection, however, BLI uses a learned amortized estimator trained for entropy modeling and reconstruction quality.

3.2 BLI Codec Architecture

Unlike previous trit-plane codecs (Lee et al., 2022; Jeon et al., 2023) that reuse the initial Gaussian parameters $(\mathbf{M}^0, \mathbf{\Sigma}^0)$ throughout progressive decoding, BLI performs step-wise autoregressive refinement. Each trit-plane is divided into multiple coding steps, and both the Gaussian parameters $(\mathbf{M}^k, \mathbf{\Sigma}^k)$ and the latent estimate $\hat{\mathbf{Y}}^{k-1}$ are inferred within the current feasible bounds at step k.

Before entropy coding at step k, the BLI module operates within the feasible interval $\mathcal{I}^{k-1} = [\mathbf{L}^{k-1}, \mathbf{U}^{k-1}]$ determined by the trits decoded so far. As shown in Figure 4, the BLI module receives the interval center $\mathbf{C}^{k-1}$, the half-width $\mathbf{D}^{k-1}/2$, and the hyperprior parameters $(\mathbf{M}^0, \mathbf{\Sigma}^0)$ as inputs. A stack of Swin attention blocks (Liu et al., 2021) extracts bound-conditioned features by modeling spatial dependencies among partially decoded latent elements. These features are then processed by CNN blocks and plane-wise MLP heads to estimate $(\mathbf{M}^k, \mathbf{\Sigma}^k)$ and $\hat{\mathbf{Y}}^{k-1}$. The plane-wise MLP heads allow the module to adapt its prediction to each trit-plane level.

For the estimation of $\mathbf{M}^k$ and $\hat{\mathbf{Y}}^{k-1}$, the CNN features are additionally concatenated with $\mathbf{C}^{k-1}$ and $\mathbf{D}^{k-1}/2$, since the interval center provides a natural anchor and the half-width specifies the allowable residual range. The outputs for $\mathbf{M}^k$ and $\hat{\mathbf{Y}}^{k-1}$ are processed through a $\tanh$ activation and rescaled by $\mathbf{D}^{k-1}/2$ around $\mathbf{C}^{k-1}$, ensuring consistency with the feasible interval $\mathcal{I}^{k-1}$. As illustrated in Figures 2 and 3, the estimates at step k are used to code the next subset of trits, after which the

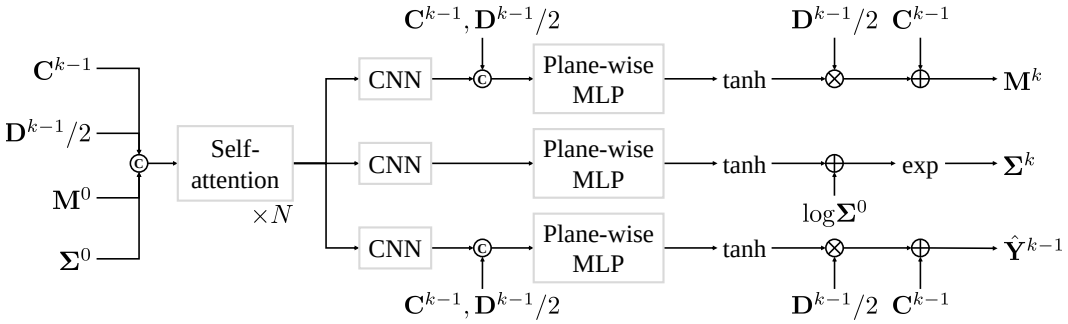


Figure 4: Architecture of the BLI module. BLI predicts the Gaussian parameters $(\mathbf{M}^k, \mathbf{\Sigma}^k)$ and the latent estimate $\hat{\mathbf{Y}}^{k-1}$ from the current bounds and hyperprior parameters.

bounds are updated to $(\mathbf{C}^k, \mathbf{D}^k)$. As the decoded trits progressively tighten the feasible bounds, this step-wise refinement improves the accuracy of entropy parameter estimation and reconstruction.

3.3 Implementation

We embed the proposed BLI framework into FTIC (Li et al., 2024), a state-of-the-art learned image codec. The original entropy model in FTIC performs channel-wise autoregressive prediction for standard fixed-rate decoding. This design is not directly compatible with trit-plane coding, where each latent element may be only partially decoded at intermediate steps. We therefore replace the channel-wise autoregressive entropy model with a non-autoregressive hyperprior variant that directly estimates the initial Gaussian parameters $(\mathbf{M}^0, \mathbf{\Sigma}^0)$ from the hyper latent tensor $\mathbf{Z}$.

Entropy coding is performed using the ANS algorithm (Duda, 2013). We train the codecs on the Flickr2W dataset (Liu et al., 2020) using 256×256 patches, a batch size of 4, and the AdamW optimizer (Loshchilov and Hutter, 2019). Training is conducted on an NVIDIA RTX A6000 GPU.

The BLI codec is optimized in two stages. First, we train the modified FTIC backbone at the highest bitrate point. It is initialized from pretrained FTIC weights and optimized for 1.0M iterations with a learning rate of 10^{-4} using the corresponding Lagrangian multiplier. This trained backbone is used as the common basis for BLI and for the DPICT (Lee et al., 2022) and CTC (Jeon et al., 2023) baselines, with the additional training scheme of Jeon et al. (2023) applied to CTC.

Second, we optimize the BLI codec using the following combined loss:

$$\mathcal{L} = \sum_{l \in \mathcal{K}_{\text{end}}} \alpha_l \Big(\sum_{k=1}^{l} \mathcal{L}_R^k + \lambda \mathcal{L}_D^l \Big), \tag{6}$$

where $\mathcal{K}_{\text{end}}$ denotes the set of endpoint step indices of the trit-planes. $\mathcal{L}_R^k$ and $\mathcal{L}_D^l$ are defined in (3) and (5), respectively. The rate term is accumulated up to each endpoint step $l \in \mathcal{K}_{\text{end}}$, so that all coding steps up to the corresponding trit-plane endpoint contribute to the loss. The weight α_l balances the contributions of endpoint losses at different trit-plane levels, because the corresponding quantization intervals have different scales; its detailed setting is provided in Appendix B.2. We set $\lambda = 0.0483$ and optimize BLI for 0.5M iterations with a learning rate of 5×10^{-5}.

For transmission scheduling, we use random trit assignments during training and the R-D prioritized transmission order of Lee et al. (2022) during evaluation. During training, each trit-plane is divided into four steps, *i.e.*, $\mathcal{K}_{\text{end}} = \{4, 8, 12, 16, 20\}$, with trits randomly assigned across steps to expose BLI to diverse partial-observation patterns. During evaluation, we use $\mathcal{K}_{\text{end}} = \{4, 8, 14, 30, 54\}$, since the trit-planes closer to the LST typically contain more high-entropy trits and therefore benefit more from finer subdivision. Further implementation details are provided in Appendix B.

4 Experimental Results

We evaluate the proposed BLI codec on the Kodak Lossless dataset (Kodak, 1999), the CLIC 2021 validation set (CLIC, 2021), and the JPEG-AI test set (JPEG-AI, 2020). Kodak contains 24 images at

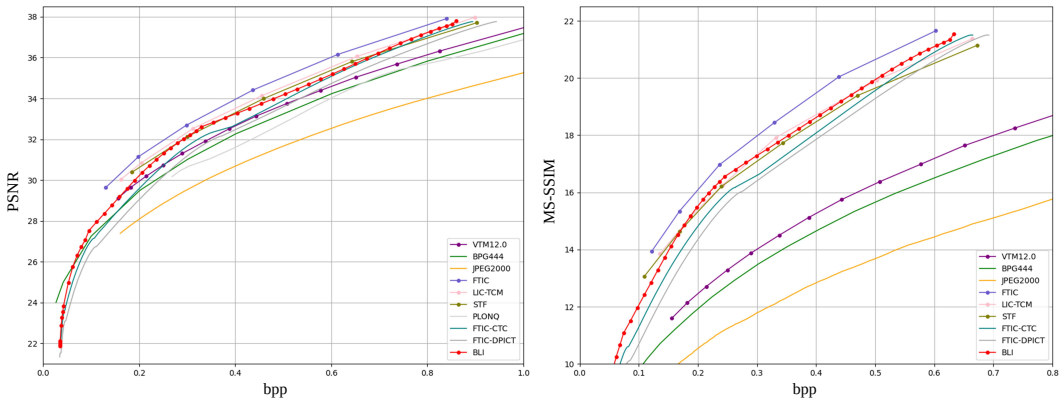


Figure 5: R-D curves on Kodak against traditional, fixed-rate learned, and progressive learned codecs. The left plot reports PSNR, and the right plot reports MS-SSIM in dB.

Table 1: PSNR BD-rates (%) with respect to FTIC-DPICT.

		Kodak	CLIC	JPEG-AI
Traditional	JPEG2000 (Skodras et al., 2001)	47.54	58.41	55.73
	BPG444 (Bellard, 2014)	−7.92	0.21	0.74
	VTM 12.0 (Bross et al., 2021)	−2.14	−2.17	−0.85
Fixed-rate	STF (Zou et al., 2022)	−14.85	−12.87	−11.69
	LIC-TCM (Liu et al., 2023)	−18.69	−19.37	−18.18
	FTIC (Li et al., 2024)	−26.37	−23.33	−20.49
Progressive	PLONQ (Lu et al., 2021)	14.83	-	21.29
	FTIC-CTC (Jeon et al., 2023)	−9.82	−11.52	−11.53
	BLI	−17.78	−17.12	−17.52

Table 2: Comparison of parameter counts and latencies. The channel-wise autoregressive entropy model of FTIC is removed for progressive methods, and the DPICT postprocessing network is not used.

		# Params. (M)	Latency (s)
Fixed-rate	LIC-TCM (Liu et al., 2023)	76.57	6.53
	FTIC (Li et al., 2024)	71.01	215.73
Progressive	FTIC-DPICT (Lee et al., 2022)	40.53	5.85
	FTIC-CTC (Jeon et al., 2023)	1,032.65	6.62
	BLI	196.12	4.77

resolutions of 512×768 or 768×512, while CLIC and JPEG-AI contain 41 and 16 higher-resolution images, respectively. Compression performance is measured in bits per pixel (bpp), PSNR, and MS-SSIM, and summarized using BD-rate (Pateux and Jung, 2007).

4.1 Comparative Assessment

R-D curve comparison: We compare BLI with representative image compression methods: JPEG2000 (Skodras et al., 2001), BPG444 (Bellard, 2014), and VTM 12.0 (Bross et al., 2021) as traditional codecs; STF (Zou et al., 2022), LIC-TCM (Liu et al., 2023), and FTIC (Li et al., 2024) as fixed-rate learned codecs; and PLONQ (Lu et al., 2021), DPICT (Lee et al., 2022), and CTC (Jeon et al., 2023) as progressive codecs. Since earlier progressive methods were built on CNN-based baselines (Minnen et al., 2018; Cheng et al., 2020), we re-implement DPICT and CTC on the stronger FTIC backbone for fair comparison, denoted as FTIC-DPICT and FTIC-CTC. Figure 5 shows the PSNR and MS-SSIM R-D curves on the Kodak dataset, with MS-SSIM converted to dB. BLI consistently outperforms existing progressive codecs over a wide bitrate range, with particularly strong gains at low-to-medium bitrates, where limited decoded information makes bound-conditioned inference especially important for reducing reconstruction distortions. Additional R-D curves and BD-rate results on the CLIC and JPEG-AI datasets are provided in Appendix A.

Compared with fixed-rate learned codecs, BLI remains competitive despite the additional constraint of progressive decoding. In particular, BLI approaches the performance of strong fixed-rate methods and surpasses STF over most of the MS-SSIM range. This suggests that bound-conditioned refinement helps preserve structural and perceptual details. Note that STF, LIC-TCM, and FTIC are individually optimized for each target rate, whereas BLI generates all R-D points from a single progressive codec.

Among progressive approaches, BLI achieves the best R-D performance over most of the bitrate range. Its consistent advantage over FTIC-DPICT reflects the benefit of bound-conditioned inference. The gains over FTIC-CTC arise from finer step-wise autoregressive modeling within each trit-plane, which enables more effective use of spatial context and bound information for entropy modeling and latent reconstruction.

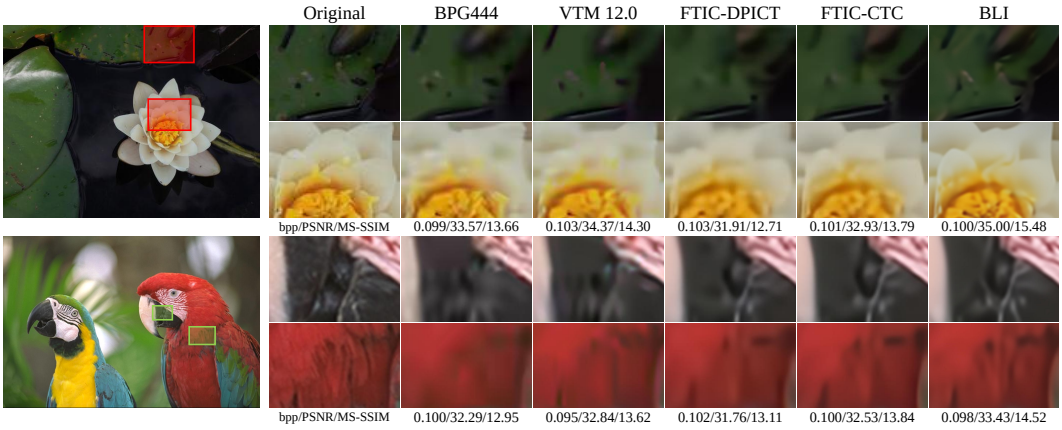


Figure 6: Qualitative comparison against representative traditional and progressive codecs.

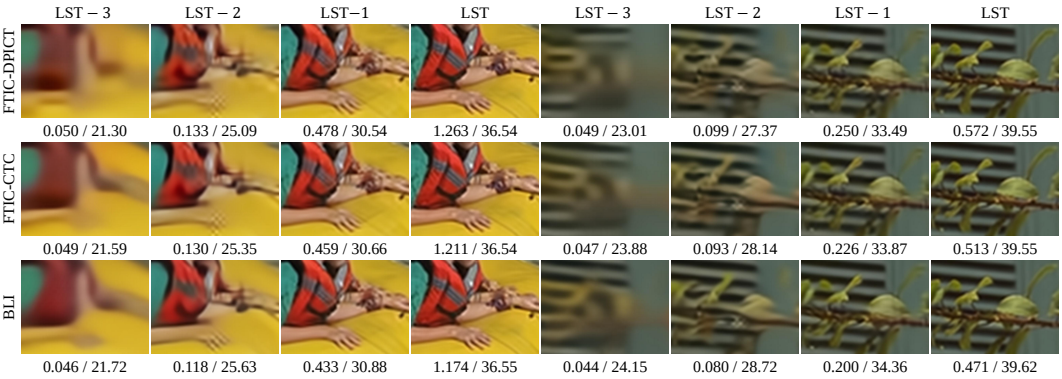


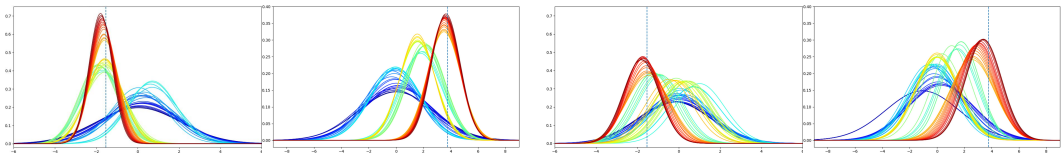
Figure 7: Qualitative comparison with existing trit-plane codecs at trit-plane endpoints. Within each column, the same number of trits is transmitted for all three codecs. The numbers below each result indicate bpp and PSNR.

BD-rate analysis: Table 1 reports PSNR BD-rates relative to FTIC-DPICT. BLI reduces bitrate by 17.78%, 17.12%, and 17.52% on Kodak, CLIC, and JPEG-AI, respectively, showing consistent gains within the progressive setting. These improvements stem from flexible in-plane autoregression, which exploits spatial context and bound information at multiple coding steps. In contrast, FTIC-CTC does not perform step-wise bound-conditioned refinement within each trit-plane, limiting its coding efficiency. Overall, BLI provides a clear R-D advantage for progressive image compression.

Model complexity: Table 2 compares the parameter counts and latency of BLI with fixed-rate and progressive baselines. BLI has about $5.3\times$ fewer parameters than FTIC-CTC, which uses a separate prediction network for each trit-plane and therefore incurs substantial parameter overhead. Although BLI requires more parameters than FTIC-DPICT, it replaces plane-specific predictors with a shared BLI module and achieves the lowest latency among the compared codecs. This latency reduction stems from efficient Gaussian interval integration, which lowers non-neural computation compared with the loop-based discrete probability composition used in FTIC-DPICT and FTIC-CTC.

Qualitative results: Figure 6 compares reconstructed images at nearly identical bitrates. Traditional codecs such as BPG444 and VTM 12.0 introduce noticeable blurring around petal boundaries and feather textures. FTIC-DPICT also suffers from blurred details, while FTIC-CTC produces sharper structures but still fails to recover fine textures faithfully. In contrast, BLI better preserves high-frequency details and produces fewer noticeable artifacts overall. This improvement comes from BLI's bound-conditioned autoregressive refinement, which uses spatial context and interval constraints to recover fine structures from partially decoded latents.

Figure 7 compares BLI with existing trit-plane codecs at the endpoints of trit-planes, where the same number of trits is transmitted within each column. BLI consistently achieves the best R-D



(a) BLI (b) Method I

Figure 8: Step-wise updates to predicted Gaussian distributions for two latent elements during evaluation. We compare BLI with method I, the ablation variant without the bound-width input and interval-rescaled output constraint. Colors transition from blue to red as decoding proceeds.

Table 3: Comparison of ablation variants and per-plane refinement schedules on the Kodak dataset. BD-rates are measured relative to FTIC-DPICT.

	# Params. (M)	Latency (s)	BD-rate (%)
Method I	155.28	4.74	−15.64
Method II	139.41	4.58	−15.01
Method III	119.32	4.00	−12.48
1 step/plane	196.12	3.82	−10.40
2 steps/plane	196.12	3.92	−12.90
4 steps/plane	196.12	4.15	−14.79
BLI	196.12	4.77	−17.78

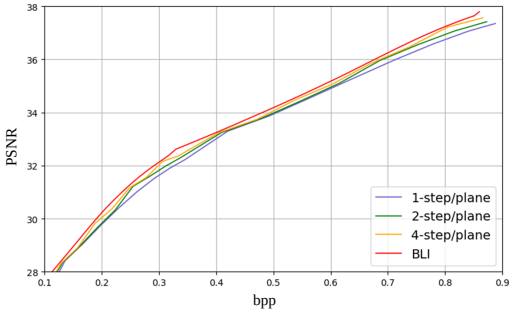


Figure 9: Effect of the number of refinement steps per trit-plane on the Kodak dataset.

performance, followed by FTIC-CTC and FTIC-DPICT. This shows that, even at the same trit-plane endpoint, BLI encodes the transmitted trits with fewer bits while producing more accurate latent estimates. The results further confirm that bound-conditioned inference improves both entropy modeling and reconstruction at intermediate progressive decoding stages.

4.2 Analysis

Model design ablations: We evaluate three variants to examine the effects of the bound-related model design, the reconstruction prediction branch, and module capacity. Table 3 summarizes the results. In method I, we remove $\mathbf{D}^{k-1}/2$ from the BLI input and disable the interval-rescaled output constraint for $\mathbf{M}^k$ and $\hat{\mathbf{Y}}^{k-1}$. In method II, we remove the reconstruction prediction branch and instead use the expectation computed from the predicted Gaussian distribution for reconstruction. In method III, we reduce the hidden dimension of the attention module and the number of CNN layers.

As shown in Table 3, method I results in a BD-rate of -15.64%, indicating that the bound width and interval-consistent prediction are important for effective progressive inference. Method II yields -15.01%, showing that directly estimating the latent reconstruction is more effective than relying on the Gaussian expectation. Method III gives -12.48%, confirming the effect of module capacity.

Step-wise distribution refinement: Figure 8 illustrates how the predicted Gaussian distributions for selected latent elements evolve during evaluation. Each curve corresponds to the predicted distribution at a particular step, with colors transitioning from blue (early steps) to red (later steps). The vertical dashed line indicates the ground-truth latent value. Compared with method I, BLI shifts the predicted distributions more consistently toward the ground-truth latent and shows clearer variance reduction over successive steps. This confirms that the bound-width input and interval-rescaled output constraint help refine uncertainty during progressive decoding.

Refinement steps per trit-plane: Table 3 and Figure 9 analyze the effect of the number of refinement steps within each trit-plane on the Kodak dataset. We compare schedules using one, two, or four steps per trit-plane with the full BLI schedule, *i.e.*, the evaluation schedule with $\mathcal{K}_{\text{end}} = \{4, 8, 14, 30, 54\}$. As the number of steps increases, R-D performance improves consistently: the BD-rate decreases from -10.40% for the 1-step setting to -12.90% and -14.79% for the 2-step and 4-step settings, respectively, and further reaches -17.78% for the full setting. Notably, even the 1-step configuration slightly outperforms FTIC-CTC (-10.40% vs. -9.82%), despite using far fewer parameters and

Table 4: Cross-backbone PSNR BD-rates (%) with respect to the corresponding DPICT baseline.

Backbone	CTC	BLI
Cheng et al. (2020)	-14.84	-19.87
Liu et al. (2023)	-6.86	-22.30

Table 5: Controlled comparison with discrete prediction. PSNR BD-rates (%) are relative to FTIC-DPICT.

Method	Params. (M)	1 step/plane	Full
IPAR + discrete	207.80	-9.30	-14.06
BLI	196.12	-10.40	-17.78

lower latency. These results show that BLI achieves strong R-D performance through more frequent in-plane autoregressive refinement, which allows bound information and spatial context to be used more effectively throughout progressive decoding.

Cross-backbone generalization: We further evaluate BLI on two backbones of learned image codecs beyond FTIC: the CNN-based backbone of Cheng et al. (2020), used in the original DPICT and CTC papers, and Liu et al. (2023). Table 4 reports BD-rates with respect to the corresponding DPICT baseline. On the backbone of Cheng et al. (2020), BLI achieves -19.87%, compared with the -14.84% reported for CTC in its original paper. On the backbone of Liu et al. (2023), we train CTC and BLI under the same training configurations. BLI improves the BD-rate from -6.86% to -22.30%. These results indicate that the benefit of BLI is preserved across different g_a and g_s backbones.

Discrete vs. continuous distributions: To distinguish the effect of BLI's continuous distribution modeling from that of in-plane autoregressive refinement, we replace Gaussian interval integration with direct ternary-logit prediction, similar to CTC, while retaining the same context module, reconstruction branch and refinement schedule. Table 5 compares this variant, denoted "IPAR + discrete", with BLI at matched refinement depths. BLI achieves lower BD-rates with one step per plane (-10.40% vs. -9.30%) and with the full schedule (-17.78% vs. -14.06%), despite using fewer parameters. Thus, increasing the number of refinement steps benefits both methods, while direct discrete prediction does not reproduce the gains of continuous interval-based distribution modeling.

4.3 Limitations and Future Work

Although BLI supports flexible transmission schedules within each trit-plane, our current evaluation adopts the R-D-prioritized transmission order of Lee et al. (2022). Designing transmission orders better matched to bound-conditioned inference remains an important direction for future work. A performance gap also remains between BLI and recent fixed-rate codecs, as expected for progressive codecs that reconstruct from bitstream prefixes. Closing this gap is another important direction for future work.

5 Conclusions

We introduced bound-conditioned latent inference (BLI), a progressive image compression framework that reformulates trit-plane coding as sequential latent inference under shrinking interval constraints. By jointly refining the Gaussian parameters for entropy modeling and the latent estimates for reconstruction within progressively updated bounds, BLI enables flexible in-plane autoregressive refinement and more effective use of the information shared by the encoder and decoder at each coding step. Experiments show that BLI achieves state-of-the-art R-D performance among progressive codecs and preserves high-frequency details more faithfully, while replacing CTC's plane-specific predictors with a shared BLI module that uses fewer parameters and achieves lower latency. Overall, BLI provides an effective framework for progressive learned image compression.

Acknowledgments and Disclosure of Funding

This work was supported by the National Research Foundation of Korea (NRF) grants funded by the Korea government(MSIT) (No. RS-2024-00397293 and No. RS-2026-25616634).

The authors declare no competing interests.

References

Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc Van Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *NeurIPS*, 2017.

Nicola Asuni and Andrea Giachetti. TESTIMAGES: a large-scale archive for testing visual devices and basic image processing algorithms. In *STAG*, 2014.

Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end optimized image compression. In *ICLR*, 2017.

Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *ICLR*, 2018.

Heinz H. Bauschke and Jonathan M. Borwein. On projection algorithms for solving convex feasibility problems. *SIAM Rev.*, 38, 1996.

F. Bellard. BPG image format. Online: `https://bellard.org/bpg`, 2014.

Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J. Sullivan, and Jens-Rainer Ohm. Overview of the versatile video coding (VVC) standard and its applications. *IEEE TCSVT*, 31, 2021.

Tong Chen and Zhan Ma. Variable bitrate image compression with quality scaling factors. In *ICASSP*, 2020.

Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized Gaussian mixture likelihoods and attention modules. In *CVPR*, 2020.

Francois Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017.

CLIC. Workshop and challenge on learned image compression. In *CVPRW*, 2021.

Ze Cui, Jing Wang, Shanyin Gao, Tiansheng Guo, Yihui Feng, and Bo Bai. Asymmetric gained deep image compression with continuous rate adaptation. In *CVPR*, 2021.

Jarek Duda. Asymmetric numeral systems: entropy coding combining speed of Huffman coding with compression rate of arithmetic coding. *arXiv preprint arXiv:1311.2540*, 2013.

Vivek K. Goyal. Theoretical foundations of transform coding. *IEEE Signal Process. Mag.*, 18, 2001.

Dailan He, Yaoyan Zheng, Baocheng Sun, Yan Wang, and Hongwei Qin. Checkerboard context model for efficient learned image compression. In *CVPR*, 2021.

Seungmin Jeon, Kwang Pyo Choi, Youngo Park, and Chang-Su Kim. Context-based trit-plane coding for progressive image compression. In *CVPR*, 2023.

JPEG-AI. JPEG-AI test images. Online: `https://jpegai.github.io/test_images`, 2020.

Jun-Hyuk Kim, Byeongho Heo, and Jong-Seok Lee. Joint global and local hierarchical priors for learned image compression. In *CVPR*, 2022.

Kodak. Kodak lossless true color image suite. Online: `https://r0k.us/graphics/kodak`, 1999.

Jae-Han Lee, Seungmin Jeon, Kwang Pyo Choi, Youngo Park, and Chang-Su Kim. DPICT: Deep progressive image compression using trit-planes. In *CVPR*, 2022.

Jooyoung Lee, Seunghyun Cho, and Seung-Kwon Beack. Context-adaptive entropy model for end-to-end optimized image compression. In *ICLR*, 2019.

Han Li, Shaohui Li, Wenrui Dai, Chenglin Li, Junni Zou, and Hongkai Xiong. Frequency-aware transformer for learned image compression. In *ICLR*, 2024.

Jiaheng Liu, Guo Lu, Zhihao Hu, and Dong Xu. A unified end-to-end framework for efficient deep image compression. *arXiv preprint arXiv:2002.03370*, 2020.

Jinming Liu, Heming Sun, and Jiro Katto. Learned image compression with mixed transformer-CNN architectures. In *CVPR*, 2023.

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 *ICCV*, 2021.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Yadong Lu, Yinhao Zhu, Yang Yang, Amir Said, and Taco S. Cohen. Progressive neural image compression with nested quantization and latent ordering. In *ICIP*, 2021.

Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Conditional probability models for deep image compression. In *CVPR*, 2018.

David Minnen and Saurabh Singh. Channel-wise autoregressive entropy models for learned image compression. In *ICIP*, 2020.

David Minnen, Johannes Ballé, and George Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *NeurIPS*, 2018.

S. Pateux and J. Jung. An Excel add-in for computing Bjontegaard metric and its evolution, 2007.

Yichen Qian, Xiuyu Sun, Ming Lin, Zhiyu Tan, and Rong Jin. Entroformer: A transformer-based entropy model for learned image compression. In *ICLR*, 2022.

Amir Said and William A. Pearlman. A new, fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE TCSVT*, 6, 1996.

Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The JPEG 2000 still image compression standard. *IEEE Signal Process. Mag.*, 18, 2001.

Myungseo Song, Jinyoung Choi, and Bohyung Han. Variable-rate deep image compression through spatially-adaptive feature transform. In *ICCV*, 2021.

Kedeng Tong, Yaojun Wu, Yue Li, Kai Zhang, Li Zhang, and Xin Jin. QVRF: A quantization-error-aware variable rate framework for learned image compression. In *ICIP*, 2023.

Gregory K. Wallace. The JPEG still picture compression standard. *IEEE Trans. Consum. Electron.*, 38, 1992.

Hao Xu, Xiaolin Wu, and Xi Zhang. Multirate neural image compression with adaptive lattice vector quantization. In *CVPR*, 2025.

Fei Yang, Luis Herranz, Yongmei Cheng, and Mikhail G. Mozerov. Slimmable compressive autoencoders for practical neural image compression. In *CVPR*, 2021.

Yinhao Zhu, Yang Yang, and Taco Cohen. Transformer-based transform coding. In *ICLR*, 2022.

Renjie Zou, Chunfeng Song, and Zhaoxiang Zhang. The devil is in the details: Window-based attention for image compression. In *CVPR*, 2022.

A Additional Experimental Results

More R-D curves: For completeness, we provide additional comparative results on the Kodak, CLIC 2021, and JPEG-AI datasets. Figure 10 presents the full PSNR and MS-SSIM R-D curves for all three datasets. Table 6 summarizes the BD-rates for both PSNR and MS-SSIM with respect to FTIC-DPICT.

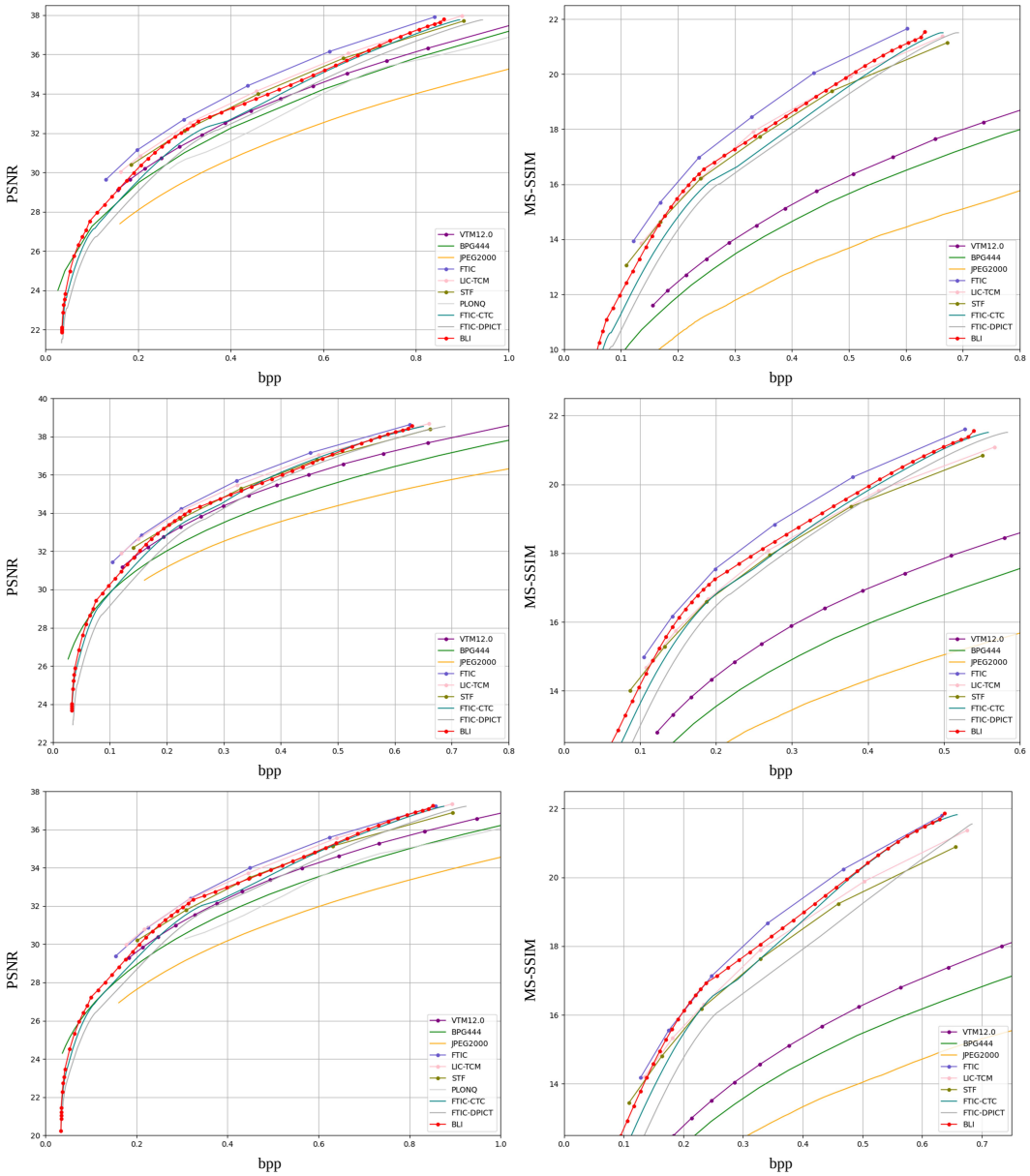


Figure 10: R-D curve comparison of the proposed BLI codec with existing progressive and fixed-rate image codecs. From top to bottom, the rows correspond to Kodak, CLIC 2021, and JPEG-AI. Best viewed digitally with zooming.

B Implementation Details

B.1 Architecture Configurations

We embed the proposed BLI framework into FTIC (Li et al., 2024), where the hyper latent $\mathbf{Z}$ has 128 channels and the primary latent $\mathbf{Y}$ has 320 channels. To estimate the initial Gaussian parameters

Table 6: BD-rates (%) with respect to FTIC-DPICT.

		PSNR			MS-SSIM		
		Kodak	CLIC	JPEG-AI	Kodak	CLIC	JPEG-AI
Traditional	JPEG2000 (Skodras et al., 2001)	47.54	58.41	55.73	183.09	232.17	205.86
	BPG444 (Bellard, 2014)	−7.92	0.21	0.74	64.25	74.67	89.49
	VTM 12.0 (Bross et al., 2021)	−2.14	−2.17	−0.85	69.49	72.06	94.68
Fixed-rate	STF (Zou et al., 2022)	−14.85	−12.87	−11.69	−12.64	−7.73	−5.57
	LIC-TCM (Liu et al., 2023)	−18.69	−19.37	−18.18	−13.84	−7.74	−6.53
	FTIC (Li et al., 2024)	−26.37	−23.33	−20.49	−25.27	−20.44	−16.58
Progressive	PLONQ (Lu et al., 2021)	14.83	-	21.29	-	-	-
	FTIC-CTC (Jeon et al., 2023)	−9.82	−11.52	−11.53	−8.43	−10.58	−9.20
	BLI	−17.78	−17.12	−17.52	−18.56	−18.14	−17.61

$(\mathbf{M}^0, \mathbf{\Sigma}^0)$, we employ two identical predictor networks. Each predictor consists of a residual CNN block with three 3×3 convolution layers, followed by two FAT blocks (Li et al., 2024), and a final 3×3 convolution layer, yielding a 320-channel output tensor that provides the initial distribution estimates. All latent-related tensors, including $\mathbf{Y}$, $\mathbf{M}^k$, $\mathbf{\Sigma}^k$, $\mathbf{L}^k$, $\mathbf{U}^k$, $\mathbf{C}^k$, and $\mathbf{D}^k$, have shape $C \times H \times W$, where $C = 320$ and (H, W) are $1/16$ of the input image resolution. In the BLI model architecture, a total of 10 Swin attention (Liu et al., 2021) layers are used, each with a hidden dimension of 960, 16 attention heads, and a window size of 8. Each CNN branch uses four depthwise convolution (Chollet, 2017) layers to improve efficiency while preserving spatial modeling capacity.

B.2 Training Loss

In (6), the endpoint indices are given by $\mathcal{K}_{\text{end}} = \{4, 8, 12, 16, 20\}$, where $l = 4$ and $l = 20$ represent the MST and the LST, respectively. The weight α_l is adopted to account for the different quantization intervals across plane levels. Specifically, because the quantization interval increases by a factor of 3 from the LST to the MST, we set α_l proportional to the inverse squared quantization interval. As a result, the weights decrease by a factor of $1/9$ from the LST to the MST, corresponding to $\{(1/9)^0, (1/9)^1, \ldots, (1/9)^4\}$ from the LST to the MST.

B.3 Transmission Schedule

In BLI, the trits transmitted within each trit-plane are divided into multiple coding steps. The step endpoints are represented by $\mathcal{K}_{\text{end}}$, which gives the cumulative coding-step indices at the endpoints of successive trit-planes. During training, each trit-plane is uniformly divided into four steps, yielding $\mathcal{K}_{\text{end}} = \{4, 8, 12, 16, 20\}$. To expose BLI to diverse partial-observation patterns, we randomly assign trits within each trit-plane to the four steps during the initial training stage. We then fine-tune the codec using the same R-D-prioritized transmission order (Lee et al., 2022) as in evaluation.

During evaluation, we use a nonuniform full BLI schedule, $\mathcal{K}_{\text{end}} = \{4, 8, 14, 30, 54\}$, corresponding to step allocations of $(4, 4, 6, 16, 24)$ from the MST to the LST. This schedule allocates more refinement steps to trit-planes closer to the LST, since these planes typically contain more high-entropy trits and benefit more from finer in-plane subdivision. Within each trit-plane, the assignment of trits to coding steps follows the R-D-prioritized order. Specifically, for each latent element, we estimate the expected bit increase ΔR and the distortion change ΔD using the predicted distribution parameters available at the beginning of the plane. Trits are then ranked in descending order of $-\Delta D/\Delta R$, so that those expected to yield larger distortion reduction per bit are transmitted earlier.

B.4 Trit Partitioning Within a Trit-Plane

Using the initial scale parameters $\mathbf{\Sigma}^0$ predicted from the hyper latent $\hat{\mathbf{Z}}$, we determine the number of trits required to represent each latent element. For every $\mathbf{\Sigma}_i^0$, we compute the width of the symmetric Gaussian interval that contains probability mass $1 - 10^{-9}$. This interval length is given by $2\kappa\mathbf{\Sigma}_i^0$, where $\kappa = -\Phi^{-1}(5 \times 10^{-10})$. The required number of trits is then obtained by taking the ceiling of the base-3 logarithm of this length:

$$n_i = \lceil \log_3(2\kappa\mathbf{\Sigma}_i^0) \rceil. \tag{7}$$

Thus, n_i represents the number of ternary digits needed to encode the i-th latent element. In each trit-plane, before processing the first step, we partition all elements satisfying $n_i \geq l$ in the current plane l into multiple subsets, which correspond to the coding steps.

B.5 Algorithmic Workflow

Since the encoder and decoder share the same transmission schedule described in Section B.3, they can identify the trits processed at each step. Once the trits for a given step are decoded, the corresponding latent bounds are updated by shrinking to the selected sub-intervals. Full pseudo-code for the encoding and decoding procedures is provided in Algorithms 1 and 2. During training, the forward pass follows the same procedure as the decoding process in Algorithm 2, except that no entropy decoding is performed. Instead, the lower and upper bounds at each step are directly obtained from the ground-truth latent $\mathbf{Y}$.

Algorithm 1: Encoding in BLI

Data: Initial parameters $\mathbf{M}^0, \mathbf{\Sigma}^0$, trit counts n_i
Result: Encoded bitstream

`/* Initialization */`

Compute initial bounds:

$$\mathbf{C}_i^0 \leftarrow \mathbf{M}_i^0, \qquad \mathbf{D}_i^0 \leftarrow 3^{n_i}.$$

Set step index $k \leftarrow 1$;

while *not all trits encoded* **do**

- Estimate $(\mathbf{M}^k, \mathbf{\Sigma}^k, \hat{\mathbf{Y}}^{k-1})$ using $(\mathbf{C}^{k-1}, \mathbf{D}^{k-1}, \mathbf{M}^0, \mathbf{\Sigma}^0)$; `// BLI`
- Identify trits to be encoded for step k;
- Encode the trits for step k using $(\mathbf{M}^k, \mathbf{\Sigma}^k)$; `// Entropy encoding`
- Update bounds $(\mathbf{C}^k, \mathbf{D}^k)$; `// Bound update`
- $k \leftarrow k + 1$;

Algorithm 2: Decoding in BLI

Data: Initial parameters $\mathbf{M}^0, \mathbf{\Sigma}^0$, trit counts n_i, encoded bitstream
Result: Reconstructed latent

`/* Initialization */`

Compute initial bounds:

$$\mathbf{C}_i^0 \leftarrow \mathbf{M}_i^0, \qquad \mathbf{D}_i^0 \leftarrow 3^{n_i}.$$

Set step index $k \leftarrow 1$;

while *target quality or bitrate not reached* **do**

- Estimate $(\mathbf{M}^k, \mathbf{\Sigma}^k, \hat{\mathbf{Y}}^{k-1})$ using $(\mathbf{C}^{k-1}, \mathbf{D}^{k-1}, \mathbf{M}^0, \mathbf{\Sigma}^0)$; `// BLI`
- Identify trits to be decoded for step k;
- Decode the trits for step k from the bitstream using $(\mathbf{M}^k, \mathbf{\Sigma}^k)$; `// Entropy decoding`
- Update bounds $(\mathbf{C}^k, \mathbf{D}^k)$ using decoded trits ; `// Bound update`
- $k \leftarrow k + 1$;

B.6 Baseline Implementations

For the FTIC (Li et al., 2024) backbone, which underlies FTIC-DPICT (Lee et al., 2022), FTIC-CTC (Jeon et al., 2023), and BLI, we replaced the original entropy model with a simple structure described

in Section B.1 and initialized the network using the pretrained FTIC weights released in its official GitHub repository for training.

We implement the traditional codecs JPEG2000 (Skodras et al., 2001), BPG444 (Bellard, 2014), and VTM 12.0 (Bross et al., 2021) as follows.

JPEG2000: We employ the open-source implementation provided in Skodras et al. (2001). The commands used for encoding and decoding are provided below. We convert the RGB images into raw format before processing.

```
{buildpath}/opj_compress -i {inputfile}
-o {bin} -r {15:150}
-F {width},{height},3,8,u@1x1:1x1:1x1

{buildpath}/opj_decompress -i {bin} -o
{outputfile}
```

BPG444: We use the software provided in Bellard (2014) and run the commands listed below.

```
{buildpath}/bpgenc {inputfile} -o {bin}
-q {26:52}
-f 444 -e x265

{buildpath}/bpgdec -o {outputfile} {bin}
```

VTM 12.0: We run the VTM 12.0 reference software package available at `https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/tree/VTM-12.0` using the following commands.

```
{buildpath}/EncoderApp -i {inputfile} -c
{cfgpath}/encoder_intra_vtm.cfg
-o /dev/null -b {bin} -wdt {width} -hgt
{height} -fr 1 -f 1
-q 24, 26, 30, 31:43
--InputChromaFormat=444
--InputBitDepth=8
--ConformanceWindowMode=1
--InputColourSpaceConvert=RGBtoGBR
--SNRInternalColourSpace=1
--OutputInternalColourSpace=0

{buildpath}/DecoderApp -b {bin} -o
{outputfile} -d 8
--OutputColourSpaceConvert=GBRtoRGB
```

For the other methods in comparison, we use their official implementations or the reported results provided in their papers.

C Extended Analyses and Results

C.1 Computational Complexity

To further analyze the computational cost of the progressive codecs, we report the floating-point operations and decompose the end-to-end latency into neural-network inference and non-neural processing. The FLOP counts are measured for a 768×512 input, corresponding to the Kodak resolution. Latency is measured on the Kodak dataset using an AMD Ryzen 9 3900X CPU and an NVIDIA RTX 4090 GPU. The non-neural processing time includes probability processing and entropy coding outside the neural-network forward passes.

BLI requires more neural computation than FTIC-DPICT because it repeatedly refines the entropy distribution and latent estimate through in-plane autoregressive inference. Its FLOP count is comparable to that of FTIC-CTC, while its neural inference time is substantially higher than those of both

Table 7: Comparison of computational complexity on the Kodak dataset.

Method	FLOPs (T)	Neural inference (s)	Non-neural processing (s)	Total latency (s)
FTIC-DPICT	41.42	0.08	5.77	5.85
FTIC-CTC	66.69	0.18	6.44	6.62
BLI	65.96	3.72	1.05	**4.77**

Table 8: Effect of the within-plane transmission ordering on the Kodak dataset. BD-rates (%) are measured relative to FTIC-DPICT.

Trans. ordering	Full BD-rate (%)	Endpoint BD-rate (%)
Random	-10.01	-19.63
Raster scan	-7.62	-18.52
σ ascending	-2.47	-18.02
R-D prioritized	$\mathbf{-17.78}$	-19.59

FTIC-DPICT and FTIC-CTC. Nevertheless, BLI achieves the lowest end-to-end latency because its non-neural processing overhead is significantly smaller.

This difference mainly arises from the computation of trit probabilities. FTIC-DPICT and FTIC-CTC construct progressive discrete probabilities using loop-based probability composition, whereas BLI directly obtains the ternary probabilities by integrating the predicted Gaussian distribution over the corresponding sub-intervals. Consequently, the additional neural computation introduced by the BLI refinement steps is offset by substantially lower non-neural processing overhead, resulting in the lowest overall latency among the three progressive codecs.

C.2 Sensitivity to Transmission Ordering

We further analyze the effect of the within-plane transmission ordering on the progressive R-D performance of BLI. Using the same trained BLI model, we compare the R-D-prioritized ordering used in our main experiments with three alternative strategies: random ordering, raster scan, and ascending order of the predicted scale parameter. Table 8 reports two types of BD-rate. The full BD-rate is computed using all progressive reconstruction points, whereas the endpoint BD-rate is computed using only the fully decoded trit-plane endpoints.

The transmission ordering has a substantial effect on the full progressive R-D performance, while its effect on the trit-plane endpoints is relatively small. The results indicate that transmission ordering mainly affects how efficiently information is delivered at intermediate prefixes within each trit-plane, while the reconstruction quality at fully decoded trit-plane endpoints is less sensitive to the ordering. The R-D-prioritized strategy therefore improves progressive efficiency primarily by transmitting high-impact trits earlier within each plane.

C.3 Entropy Modeling vs. Latent Reconstruction

To assess the relative contributions of entropy modeling and latent reconstruction, we perform test-time ablations using the same trained BLI model. We selectively disable each refinement pathway while retaining the other. Table 9 summarizes the results on the Kodak dataset.

When the step-wise refined entropy parameters $(\mathbf{M}^k, \mathbf{\Sigma}^k)$ are retained but the predicted latent estimate $\hat{\mathbf{Y}}^{k-1}$ is replaced by the Gaussian expectation, the BD-rate changes from -17.78% to -16.38%. In contrast, disabling entropy refinement and using the initial hyperprior parameters $(\mathbf{M}^0, \mathbf{\Sigma}^0)$ for entropy coding, while retaining the predicted latent reconstruction, reduces the BD-rate gain to -10.24%. These results indicate that progressively refining the entropy parameters provides a larger contribution to the overall R-D improvement. The explicit latent-reconstruction prediction provides a smaller but consistent additional gain, showing that the two pathways play complementary roles in BLI.

Table 9: Test-time ablation of entropy modeling and latent reconstruction on the Kodak dataset. BD-rates (%) are measured relative to FTIC-DPICT.

Test-time variant	Entropy params. for coding	Latent recon.	BD-rate (%)
Recon. with Gaussian mean	$(\mathbf{M}^k, \mathbf{\Sigma}^k)$	Gaussian mean	-16.38
No entropy refinement	$(\mathbf{M}^0, \mathbf{\Sigma}^0)$	$\hat{\mathbf{Y}}^{k-1}$	-10.24
Full BLI	$(\mathbf{M}^k, \mathbf{\Sigma}^k)$	$\hat{\mathbf{Y}}^{k-1}$	$\mathbf{-17.78}$

Table 10: Comparison of perceptual quality on the Kodak dataset at low bitrates. Lower LPIPS and DISTS values indicate better perceptual quality.

bpp	Metric	BPG444	VTM 12.0	FTIC-DPICT	FTIC-CTC	BLI
0.10	LPIPS	0.3848	0.3494	0.4008	0.3465	**0.3203**
	DISTS	0.2038	0.1994	0.2308	0.2089	**0.1910**
0.15	LPIPS	0.3207	0.3064	0.3569	0.3123	**0.2868**
	DISTS	0.1771	0.1800	0.2049	0.1884	**0.1736**
0.20	LPIPS	0.2760	0.2663	0.3045	0.2661	**0.2366**
	DISTS	0.1586	0.1606	0.1780	0.1630	**0.1510**

C.4 Perceptual Evaluation

We further evaluate perceptual quality at low bitrates on the Kodak dataset using LPIPS and DISTS. For each method, the metric values at common target bitrates of 0.10, 0.15, and 0.20 bpp are obtained by linear interpolation between adjacent measured operating points. Table 10 reports the dataset-level averages over all Kodak images, where lower values indicate better perceptual quality.

BLI achieves the lowest LPIPS and DISTS values at all three target bitrates, consistently outperforming both conventional codecs and the progressive learned baselines. These results indicate that the R-D improvements of BLI are also accompanied by improved perceptual quality in the low-bitrate regime, rather than being limited to distortion metrics such as PSNR and MS-SSIM.

C.5 Reconstructed Images

Figures 11$\sim$19 present examples of images progressively reconstructed by the PSNR-trained BLI model on the Kodak (Kodak, 1999), CLIC (CLIC, 2021), and JPEG-AI (JPEG-AI, 2020) datasets.

C.6 Tests on Challenging Color Patterns

To evaluate the robustness of BLI on challenging color patterns, we apply our framework to two images from the Tecnick dataset (Asuni and Giachetti, 2014) that includes smooth low-frequency gradients and sharp chromatic edges. The resulting reconstructions are shown in Figure 20.



0.038 / 25.88 / 7.66 | 0.065 / 28.86 / 10.17 | 0.212 / 33.42 / 14.71 | 0.618 / 38.00 / 19.30

0.043 / 22.26 / 5.47 | 0.128 / 25.49 / 9.36 | 0.480 / 30.84 / 14.93 | 1.201 / 36.25 / 19.78

Figure 11: Progressively reconstructed images of 'kodim04.png' and 'kodim18.png' obtained by the PSNR-trained model. The bitrate (bpp), PSNR (dB), and MS-SSIM (dB) are reported below each image.



Figure 12: Progressively reconstructed images of 'kodim08.png' and 'kodim13.png' obtained by the PSNR-trained model. The bitrate (bpp), PSNR (dB), and MS-SSIM (dB) are reported below each image.

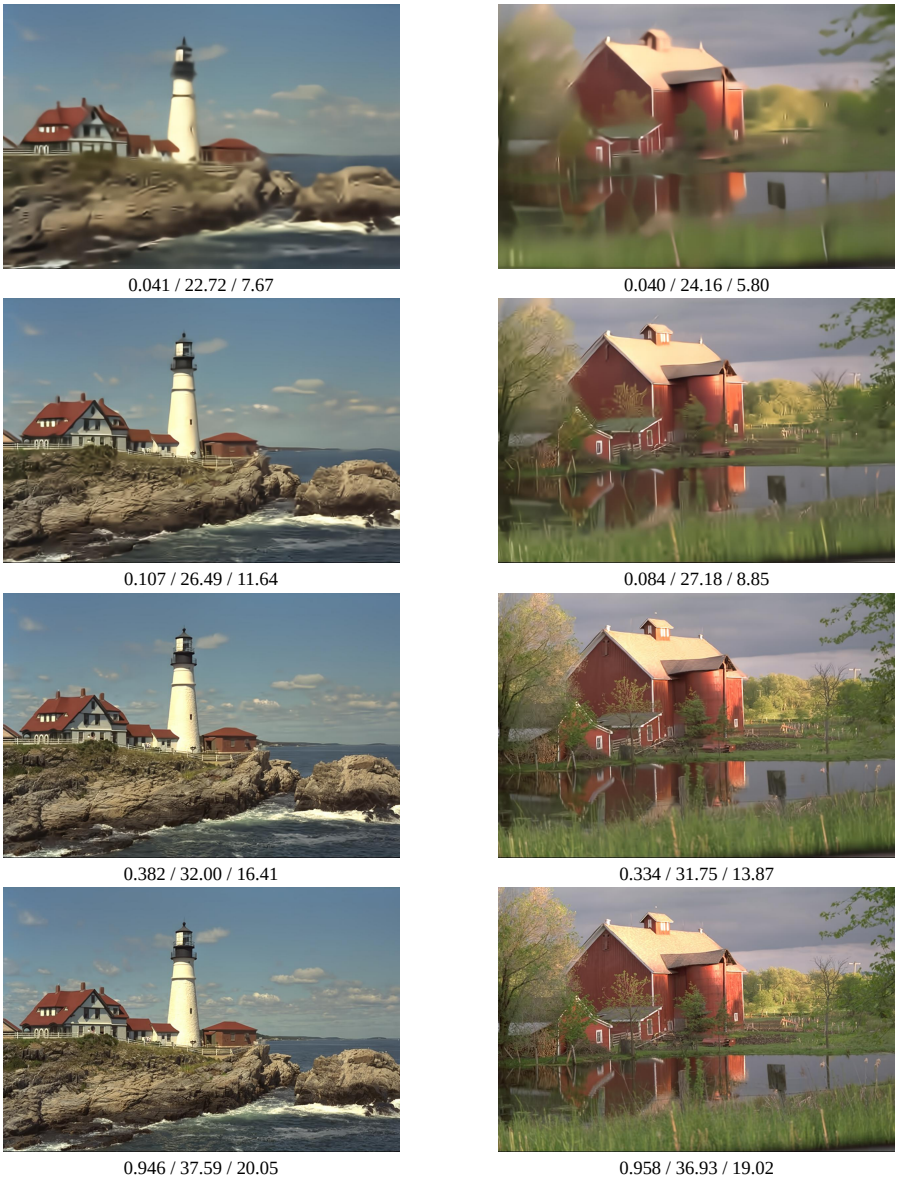


Figure 13: Progressively reconstructed images of 'kodim21.png' and 'kodim22.png' obtained by the PSNR-trained model. The bitrate (bpp), PSNR (dB), and MS-SSIM (dB) are reported below each image.



Figure 14: Progressively reconstructed images of 'andrew-ruiz-376.png' and 'clem-onojeghuo-33741.png' obtained by the PSNR-trained model. The bitrate (bpp), PSNR (dB), and MS-SSIM (dB) are reported below each image.



Figure 15: Progressively reconstructed images of 'daniel-robert-405.png' and 'mullermarc-184.png' obtained by the PSNR-trained model. The bitrate (bpp), PSNR (dB), and MS-SSIM (dB) are reported below each image.

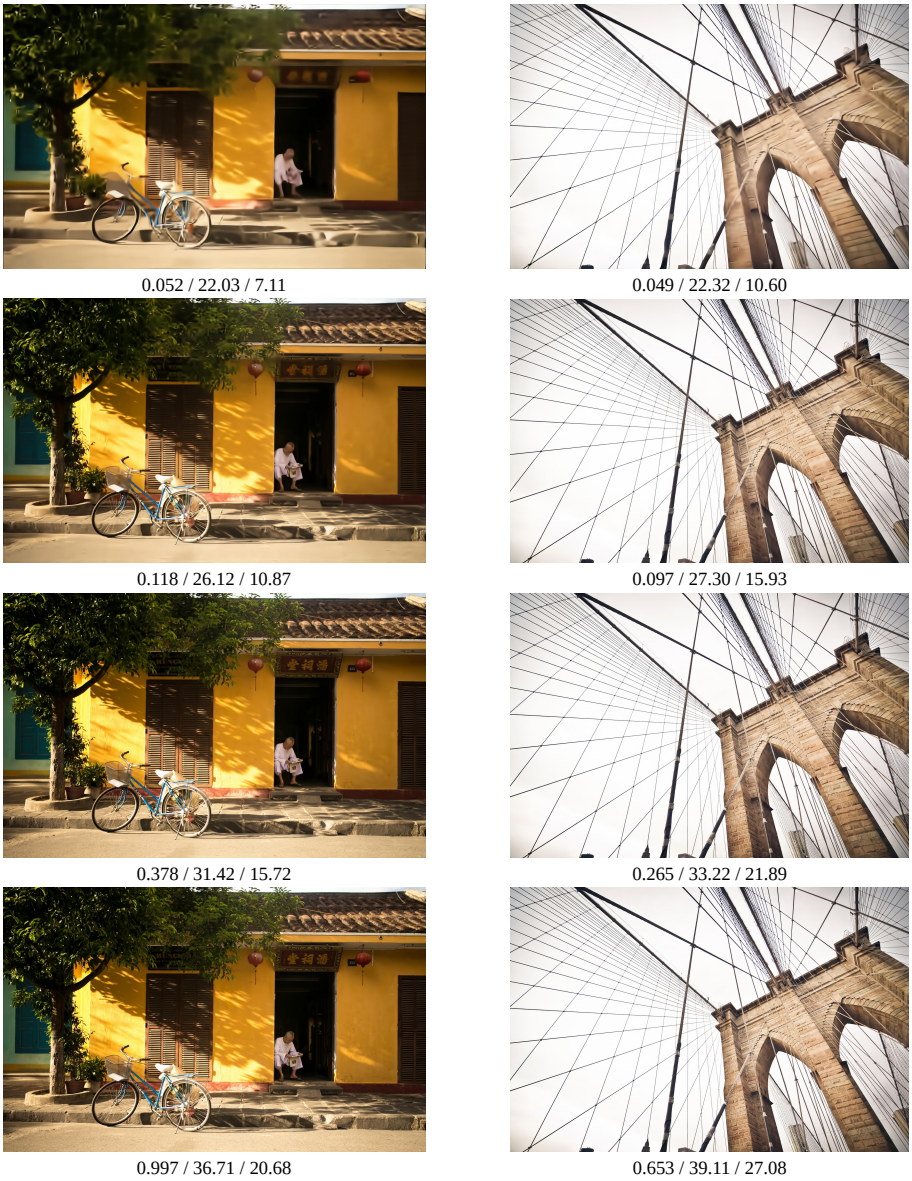


Figure 16: Progressively reconstructed images of 'thong-vo-428.png' and 'vita-vilcina-3055.png' obtained by the PSNR-trained model. The bitrate (bpp), PSNR (dB), and MS-SSIM (dB) are reported below each image.

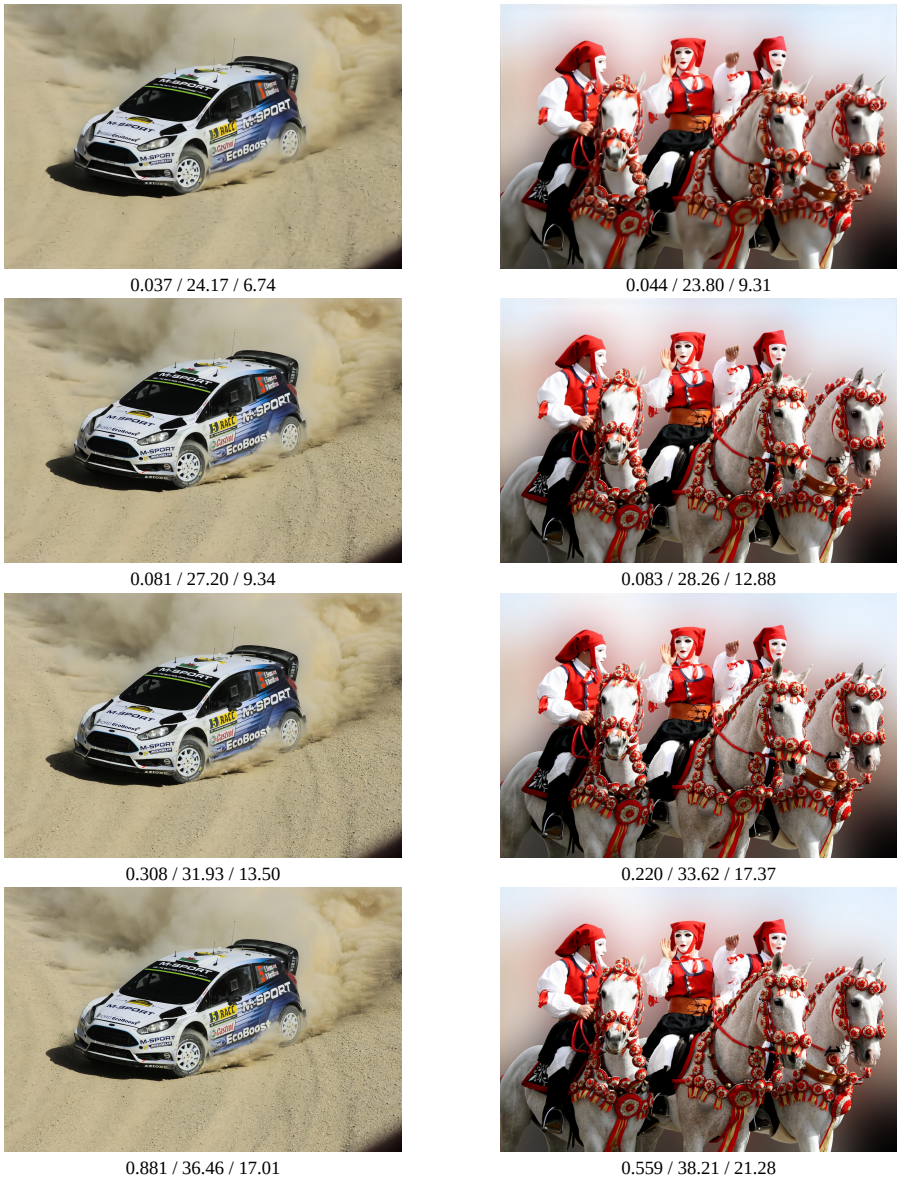


Figure 17: Progressively reconstructed images of '00002_TE_2144x1424.png' and '00003_TE_1944x1296.png' obtained by the PSNR-trained model. The bitrate (bpp), PSNR (dB), and MS-SSIM (dB) are reported below each image.



Figure 18: Progressively reconstructed images of '00005_TE_1336x872.png' and '00009_TE_1976x1312.png' obtained by the PSNR-trained model. The bitrate (bpp), PSNR (dB), and MS-SSIM (dB) are reported below each image.



Figure 19: Progressively reconstructed images of '00010_TE_2000x1128.png' and '00011_TE_1744x1160.png' obtained by the PSNR-trained model. The bitrate (bpp), PSNR (dB), and MS-SSIM (dB) are reported below each image.

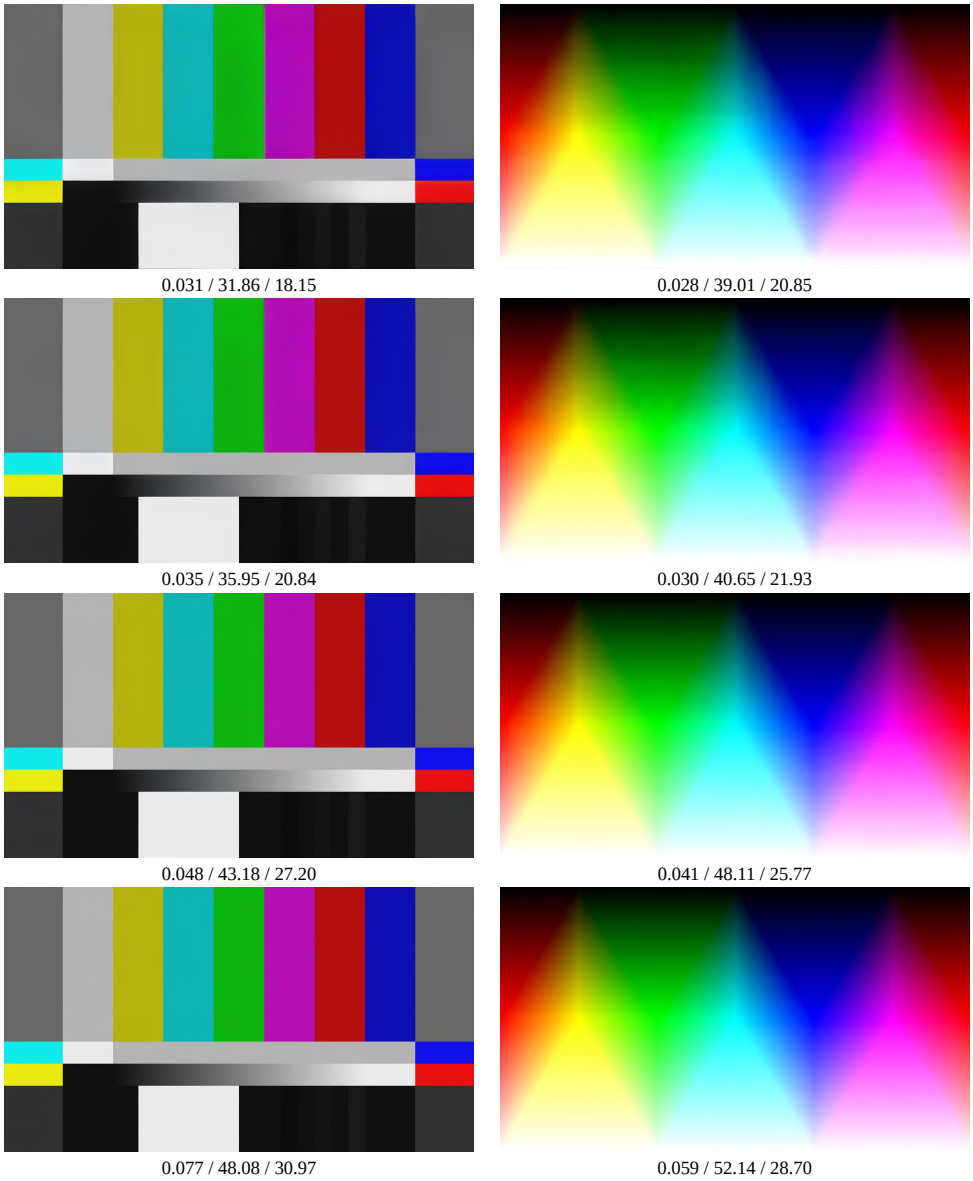


Figure 20: Progressively reconstructed images of 'img_1280x720_3x8bit_RGB_color_SMPTE_RP_219_2002.png' and 'img_1280x720_3x8bit_RGB_color_rainbow_gradient.png' obtained by the PSNR-trained model. The bitrate (bpp), PSNR (dB), and MS-SSIM (dB) are reported below each image.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims in the abstract and introduction match the paper's actual contributions and scope, including the proposed BLI framework, its bound-conditioned refinement mechanism, and its empirical advantages over prior progressive codecs.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a discussion of limitations and future work in Section 4.3.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not present theoretical results that require formal assumptions or proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides sufficient detail to reproduce the experimental results in Section 3.3 of the main paper and Section B of the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The datasets used in this work are publicly available. Our code will be released upon acceptance.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: The paper specifies the experimental settings needed to understand the results in Section 3.3 of the main paper and Section B of the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: We do not report error bars or formal statistical significance tests. Our evaluation is deterministic for a fixed trained model and benchmark set, and follows standard image compression practice.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper specifies the training hardware in Section 3.3 and reports model sizes and latency in Table 2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics, and are not aware of any aspect of this work that violates it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The paper focuses on a foundational image compression method and is not developed for a specific downstream application or deployment scenario.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper does not release high-risk models or datasets that would require special safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper credits the creators of the datasets, baselines, and backbone assets used in this work through citations in the main paper and appendix.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper does not introduce or release new assets such as datasets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in the paper does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.