SELF-CONTAINED PROBABILITY REPRESENTATION FOR DEPLOYABLE LOSSLESS IMAGE COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent learning-based lossless image compression methods achieve strong compression, but they rely on large pre-trained models whose heavy inference cost hinders practical deployment. To this end, we introduce a compact representation of image probability which is embedded in a bitstream, shortly. The representation is compiled once into a lookup table, eliminating repeated neural inference during coding. For deployable on-device encoding, we marginalize image-specific features into a reusable table without per-image fitting, while a compact tree restores self-contained delivery. Our method achieves 95.5×/52.8× faster decoding/encoding than the fastest evaluated learned baseline, and 4×/5.7× faster decoding/encoding than classical codecs, while maintaining competitive rate–throughput trade-offs. Experiments demonstrated capabilities beyond prior learned codecs, including 16-bit image compression, practical encoding and decoding on devices ranging from servers to wearables, and a lossy extension through bit-plane truncation. Self-contained decoding demo files are included in the supplementary material.

1 INTRODUCTION

Since autoregressive image modeling was introduced by van den Oord et al. (2016a;b), it has become a major paradigm for learned lossless compression, leading to numerous network-based methods with strong compression performance (Zhang et al., 2024b; Li et al., 2025). At the extreme of autoregressive prediction, Deletang et al. (2024); Chen et al. (2025); Du et al. (2026) demonstrate that large language models achieve strong image compression, consistent with established scaling laws (Henighan et al., 2020). However, deploying a practical codec requires more than a small compressed payload, as model storage and repeated inference also impose deployment costs. As summarized in Fig. 1(a), we focus on two major deployment bottlenecks: (1) repeated neural inference for probability estimation, and (2) the need for pre-trained models at both the encoder and decoder.

Several prior studies (Mentzer et al., 2019; Kang et al., 2022; Kamisli, 2023; Zhao et al., 2026; Zhang et al., 2024a) have substantially reduced the first bottleneck through hierarchical, parallel, and inference-efficient designs. Dependence on shared pre-trained models, however, remains common in learned lossless compression. Zhang et al. (2025) introduce per-image fitting for learned lossless compression, while retaining a pre-trained autoregressive prior. Per-image neural representations have also been explored for compression, including lossless coding, but practical coding efficiency remains a challenge (Dupont et al., 2022; Han et al., 2025; Strümpler et al., 2022; Guo et al., 2023).

Building on these advances, we introduce Image Probability Representation (IPR), a compact representation of the conditional probabilities used for entropy coding. For image-specific coding, a compact neural representation is embedded in a bitstream, enabling self-contained decoding without a pre-trained model (Fig. 1(b)). To obtain an enumerable coding domain, we map the image through fixed reversible transforms and bit-plane decomposition. With binary coding symbols, each conditional probability mass function (PMF) is parameterized by a single Bernoulli probability, while a restricted causal context keeps the input state finite and enumerable. To capture spatial variation while preserving enumerability, we introduce a compact Haar-based positional code with finite quantization. The resulting probability function is exhaustively evaluated and compiled once into a lookup table (LUT), removing repeated neural inference from the coding loop.

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.

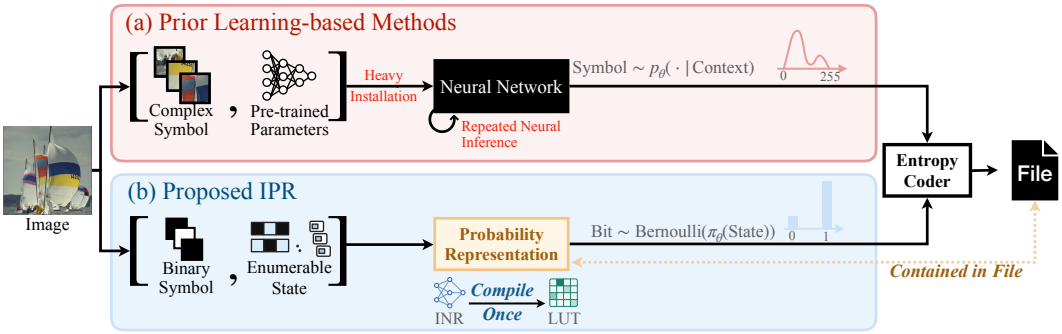


Figure 1: **Motivation for self-contained probability representation.** Conventional learned codecs rely on shared pre-trained models and repeated neural inference for probability estimation. The image-specific IPR embeds a compact probability representation in the bitstream and compiles it once into a lookup table for coding.

Image-specific fitting, however, introduces substantial encoding cost because of per-image optimization. To remove this overhead, we marginalize out image identity and spatial conditioning to obtain a reusable probability table. The table combines the model-derived marginal prior with pooled symbol counts over reference images. The resulting deployment form enables fast coding with a compact lookup table, making on-device deployment practical. For self-contained deployment, we further specialize the reusable representation with a compact Minimum Description Length (MDL) tree stored in the bitstream, removing the lookup-table dependency at the decoder. The two IPR forms target different deployment settings. IPR-I targets server-to-device delivery with an image-specific representation stored in the compressed file, while IPR-L targets repeated local coding with a reusable LUT and uses a compact MDL tree when self-contained transmission is required.

On our GPU-server benchmark, the reusable form achieves 95.5×/52.8× faster decoding/encoding than the fastest re-evaluated learned baseline at 3.19 bits per subpixel on Kodak (Franzen, 1999), while the image-specific form achieves 59.6× faster decoding at 3.14 bits per subpixel, including transmitted representations and headers.

In summary, our main contributions are as follows:

- We introduce Image Probability Representation (IPR), a compact representation of the conditional probabilities used for entropy coding, removing repeated neural inference from the coding loop by lookup-table compilation. The image-specific form enables self-contained lossless coding without a pre-trained model.
- We derive a reusable probability table by marginalizing image identity and spatial conditioning, eliminating per-image optimization for fast on-device coding. A compact MDL tree removes the shared-table dependency when self-contained deployment is required.
- We demonstrate competitive lossless compression with substantial encoding and decoding acceleration across hardware platforms ranging from servers to wearable devices. The representation extends to 16-bit images and lossy coding through bit-plane truncation.

2 Related Work

Learned Lossless Image Compression Recent learned lossless codecs improve practical coding efficiency through hierarchical prediction, parallel execution, lightweight modeling, and model specialization. Mentzer et al. (2019) use hierarchical latent representations and coarse-to-fine probability estimation to reduce sequential prediction. Kang et al. (2022) redesign learned lossless coding around GPU-oriented parallelism for practical end-to-end throughput. Motivated by the per-image overfitting strategy of Ladune et al. (2023), Zhang et al. (2025) propose per-image optimization of latent variables and network parameters while retaining a pre-trained autoregressive prior. Zhao et al. (2026) pursue lightweight LLM-based probability modeling in a unified framework across multiple data modalities. For source representation, Zhang et al. (2024b) exploit bit-plane slicing to model binary source structure with dimension-specific autoregression. Zhang et al. (2024a) compile a finite-state prior over learned discrete latents, whereas IPR represents the coding probabilities of source-derived binary symbols directly.

Neural inference nevertheless remains part of the compression–throughput trade-off. Figure 2 compares compression ratio, defined as $8/\text{bpsp}$, against end-to-end round-trip throughput from image to compressed file and back to image. The IPR variants occupy different operating points, with IPR-I favoring compression and IPR-L favoring throughput. IPR compiles the source-symbol probability function into lookup form, removing repeated neural inference from the coding loop.

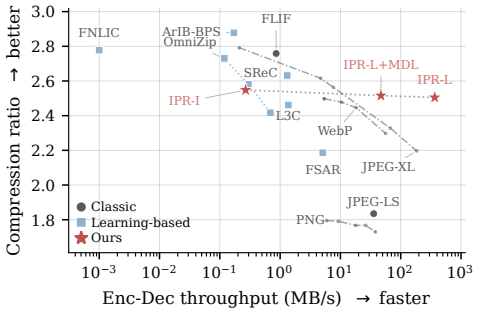


Figure 2: **Compression ratio versus end-to-end throughput** with lossless image compression baselines.

Implicit Neural Representations for Compression Implicit Neural Representations (INRs) represent signals as neural functions that map coordinates directly to signal values, enabling compression by storing the network parameters instead of the signal itself. INR-based compression was first applied to lossy image compression by Dupont et al. (2021) and subsequently extended to image and video compression (Dupont et al., 2022; Strümpler et al., 2022; Guo et al., 2023; He et al., 2023; Chen et al., 2021). It has also been adapted to exact reconstruction through bit-plane decomposition (Han et al., 2025; Tang et al., 2026). Across these approaches, the neural representation is used to estimate source values. IPR instead estimates coding probabilities, so approximation affects code length rather than the reconstructed values, while entropy decoding remains exact.

3 Method

Problem Formulation Let X denote an image and $\mathbf{x} = (x_1, \ldots, x_N)$ its sequence of source symbols. Autoregressive image models parameterize the distribution of $\mathbf{x}$ with trainable parameters θ and factorize it as

$$p_\theta(\mathbf{x}) = \prod_{i=1}^{N} p_\theta(x_i \mid x_{<i}), \tag{1}$$

where $x_{<i} = (x_1, \ldots, x_{i-1})$ denotes the causal context (van den Oord et al., 2016a). For an 8-bit source symbol, $x_i \in \{0, \ldots, 255\}$, so each conditional distribution requires a 256-way PMF. As the causal context grows, direct enumeration of $p_\theta(x_i \mid x_{<i})$ quickly becomes impractical. When p_θ is implemented by a neural probability model, the parameters θ are too large to transmit with each compressed file, and evaluating $p_\theta(x_i \mid x_{<i})$ requires repeated neural inference during coding.

3.1 Bit-Plane Representation and Reversible Transform

We therefore reformulate Eq. 1 over a reversible binary representation. Bit-plane decomposition reduces each coding symbol to a binary variable, so each conditional distribution is specified by a single Bernoulli probability. Reversible color transform and an integer CDF 5/3 DWT are applied before bit-plane decomposition to expose more predictable conditional structure and improve compression efficiency while preserving exact reconstruction. Let $\mathcal{R}$ denote the composition of the reversible RGB-to-YCoCg-R color transform, integer CDF 5/3 DWT, and bit-plane decomposition. The resulting binary representation is

$$\mathbf{b} = \mathcal{R}(X) = (b_1, \ldots, b_M), \qquad b_i \in \{0, 1\}, \qquad \mathcal{R}^{-1}(\mathbf{b}) = X. \tag{2}$$

The induced distribution over the binary representation is factorized using the restricted causal context $\mathcal{C}_i$. Let $\pi_\theta(\mathcal{C}_i) := p_\theta(b_i = 1 \mid \mathcal{C}_i)$ denote the Bernoulli probability. Then

$$p_\theta(\mathbf{b}) = \prod_{i=1}^{M} p_\theta(b_i \mid \mathcal{C}_i) = \prod_{i=1}^{M} \pi_\theta(\mathcal{C}_i)^{b_i} \left(1 - \pi_\theta(\mathcal{C}_i)\right)^{1-b_i}. \tag{3}$$

To make the probability function enumerable, we restrict $\mathcal{C}_i$ to a compact set of causal conditioning variables. Combined with binary coding, $\mathcal{C}_i$ reduces the original probability mapping to

$$\underbrace{x_{<i} \longmapsto p_\theta(x_i \mid x_{<i})}_{\text{large context space and 256-way PMF}} \quad \Longrightarrow \quad \underbrace{\mathcal{T}_b[\mathcal{C}_i] \triangleq \pi_\theta(\mathcal{C}_i)}_{\text{LUT: compact context space and scalar probability}}, \tag{4}$$

where T_b denotes the corresponding lookup table.

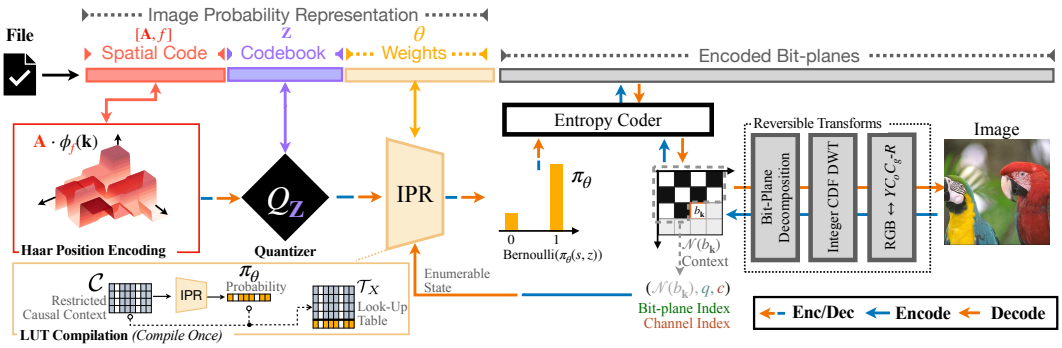


Figure 3: **Image-specific probability representation for lossless coding.** Haar positional encoding provides a compact image-specific spatial code. LUT compilation tabulates Bernoulli probabilities for direct lookup during entropy coding. The bitstream contains the selected A, f for Haar Representation (u), codebook ($\mathcal{Z}$), and network parameters (θ) together with the entropy-coded bit-planes.

To construct $\mathcal{C}_i$, we first define a discrete coding state s_i shared across the IPR variants. The state consists of a binary causal neighborhood $\mathcal{N}(b_i)$, the bit-plane index q_i, and the cell index c_i:

$$s_i := (\mathcal{N}(b_i), q_i, c_i). \tag{5}$$

All components of s_i take values from fixed discrete sets, keeping the shared coding state small enough for lookup-based probability representation.

3.2 INR-Based Image Probability Representation

We first introduce the INR-based form of IPR, denoted IPR-I, which fits a compact neural representation to each image X to model the image-specific probability function π_{θ_X}.

IPR-I augments the shared coding state s_i with an image-specific spatial code z_i, yielding

$$\mathcal{C}_i = (s_i, z_i), \qquad \pi_{\theta_X}(\mathcal{C}_i) = \pi_{\theta_X}(s_i, z_i). \tag{6}$$

Haar Position Encoding Prior works (Mildenhall et al., 2020; Wang et al., 2021) show that an appropriate encoding of spatial coordinates improves INR representation. We therefore encode the spatial coordinate $\mathbf{k} \in \mathbb{R}^2$ using a compact Haar-based representation,

$$u_i = \mathbf{A}\phi_f(\mathbf{k}_i), \tag{7}$$

where $\phi_f(\mathbf{k}_i)$ denotes the Haar basis functions selected by f, and $\mathbf{A}$ is an image-specific coefficient matrix. For each image X, we fit the probability model over a candidate pool of Haar atoms and retain a compact subset of the highest-ranked atoms to form f. For a fixed basis selection f, the fitting objective is

$$(\theta_X^\star, \mathbf{A}^\star) = \arg\min_{\theta_X, \mathbf{A}} \mathrm{BCE}(\pi_{\theta_X}(s, \mathbf{A}\phi_f(\mathbf{k})), b) + \lambda\|\mathbf{A}\|_1. \tag{8}$$

Here, BCE denotes the mean binary cross-entropy over the coding events of X, and λ controls the coefficient regularization.

Quantization and Self-Contained Representation After fitting, the resulting spatial features are clustered by k-means to obtain a finite codebook $\mathcal{Z}$,

$$z_i = Q_{\mathcal{Z}}(u_i), \qquad z_i \in \mathcal{Z}. \tag{9}$$

As illustrated in Fig. 3, the IPR-I bitstream stores the selected Haar indices f, the quantized coefficients $\mathbf{A}$, the spatial codebook $\mathcal{Z}$, and the probability-model parameters θ_X, together with the entropy-coded bit-planes. The decoder reconstructs $u_i = \mathbf{A}\phi_f(\mathbf{k}_i)$ and $z_i = Q_{\mathcal{Z}}(u_i)$ from the transmitted representation, requiring no external learned model.

Lookup-Table Compilation After reconstructing the transmitted representation, the decoder evaluates $\pi_{\theta_X}(s, z)$ over the finite conditioning domain and stores the resulting probabilities in an image-specific lookup table,

$$\mathcal{T}_X[s, z] = \pi_{\theta_X}(s, z). \tag{10}$$

The table contains $|\mathcal{S}||\mathcal{Z}|$ entries, where $\mathcal{S}$ denotes the discrete state space defined in Eq. 5. The encoder and decoder construct the same table from the transmitted representation, after which entropy coding uses direct lookup without neural inference.

3.3 LOOKUP-TABLE-BASED IMAGE PROBABILITY REPRESENTATION

Reusable Probability Representation We next introduce the LUT-based form of IPR, denoted IPR-L, which removes per-image fitting by marginalizing over X and z in $\pi_{\theta_X}(s, z)$. The resulting probability is

$$p_L(b = 1 \mid s) = \mathbb{E}_{X,z|s}\left[p(b = 1 \mid s, z, X)\right] = \sum_X \sum_{z \in \mathcal{Z}} p(b = 1 \mid s, z, X)\, p(z, X \mid s). \tag{11}$$

Given a set of $\pi_{\theta_X}(s, z)$, let

$$\widetilde{\pi}_I(s) := \sum_X \sum_{z \in \mathcal{Z}} \pi_{\theta_X}(s, z)\, p(z, X \mid s) \tag{12}$$

be an approximation to the marginal probability in Eq. 11. Note that data provide binary observations for states that are visited, while Eq 12 also provides a probability for unobserved states. We therefore combine the empirical counts with the marginal prior using KT smoothing Krichevsky & Trofimov (1981),

$$\pi_L(s) = \frac{n_s^{(1)} + \alpha \widetilde{\pi}_I(s) + \frac{1}{2}}{n_s + \alpha + 1}, \tag{13}$$

where $n_s^{(1)}$ and n_s denote the number of one-valued symbols and total coding events at state s, respectively, and α controls the strength of the model-derived prior.

Self-Contained Specialization of IPR-L IPR-L requires $\mathcal{T}_L$ at the decoder. To remove this dependency, we specialize the LUT to the current image using an MDL tree transmitted in the bitstream. Each leaf ℓ groups states sharing one Bernoulli probability,

$$\pi_\ell = \frac{S_\ell^{(1)} + \alpha \bar{\pi}_{L,\ell} + \frac{1}{2}}{S_\ell^{(1)} + S_\ell^{(0)} + \alpha + 1}, \tag{14}$$

where $S_\ell^{(v)}$ counts current-image symbols with value v in leaf ℓ, $\bar{\pi}_{L,\ell}$ is the corresponding IPR-L prior, and α controls its strength. The encoder greedily splits leaves using the components of s and retains a split only when its reduction in coding length exceeds the additional tree-description cost:

$$\Delta = L_{\text{data}}(\ell; \pi_\ell) - L_{\text{data}}(\ell_L; \pi_{\ell_L}) - L_{\text{data}}(\ell_R; \pi_{\ell_R}) - \Delta L_{\text{tree}}, \tag{15}$$

where $L_{\text{data}}(\ell; \pi) = -S_\ell^{(1)} \log_2 \pi - S_\ell^{(0)} \log_2(1 - \pi)$. The transmitted tree contains the routing structure and leaf probabilities, so decoding requires neither neural inference nor the LUT $\mathcal{T}_L$. The notation used in Section 3 is summarized in Appendix Table 5.

4 EXPERIMENTS

4.1 IMPLEMENTATION DETAILS

Representation and optimization Images are partitioned into 128×128 patches. The causal neighborhood $\mathcal{N}(b_i)$ contains 13 causal bits; its detailed layout is provided in Appendix. The cell index $c_i \in \{0, \ldots, 38\}$ identifies one of 39 channel–subband pairs. The bit-plane index q_i spans the transformed bit depth, with $q_i \in \{0, \ldots, 10\}$ for the 8-bit images and $q_i \in \{0, \ldots, 17\}$ for 16-bit images. Both indices are one-hot encoded. IPR-I selects $|f| = 32$ Haar atoms from 512 candidates. The coefficient matrix $\mathbf{A} \in \mathbb{R}^{4\times 32}$ is defined per patch, producing $u_i \in \mathbb{R}^4$ that are quantized to $z_i \in \mathcal{Z} \subset \mathbb{R}^4$ with $|\mathcal{Z}| = 32$. The π_θ uses two ReLU hidden layers, with width 32 for Kodak and 64 for the remaining datasets, followed by a scalar output. IPR-I is optimized for 4,000 Adam (Kingma & Ba, 2015) steps with learning rates 2×10^{-3} for the π_θ and 10^{-2} for $\mathbf{A}$, cosine decay, and a batch size of 262,144 binary symbols. The regularization weight is $\lambda = 10^{-4}$ for the first 2,000 steps and zero thereafter, when the top-32 Haar atoms are fixed. We use $\alpha = 32$ for IPR-L and $\alpha = 8$ for IPR-L+MDL. IPR-L constructs its reusable LUT from the first eight images of each dataset. The resulting LUT is then fixed and used to evaluate the full dataset, including the eight calibration images.

Quantization and entropy coding Given the robustness of bit-plane-based lossless INRs to parameter quantization (Han et al., 2025), we quantize $\mathbf{A}$ and the weights θ to 4 and 8 bits, respectively, while storing the spatial codebook $\mathcal{Z}$ at 16-bit precision. For the 8-bit images, the dense IPR-I and IPR-L probability tables occupy 224.9 MB and 7.03 MB, respectively. Both use 16-bit integer probabilities with a byte-renormalized range coder.

Table 1: **Quantitative comparison across diverse hardware platforms.** Single-image encoding/decoding throughput and compression rate are reported for classical codecs, learned baselines, and IPR. The upper block covers GPU-accelerated platforms, while the lower block covers devices without comparable neural acceleration. (–) denotes unavailable or unsupported measurements. (†) indicates our end-to-end evaluation implementation.

	Single-image throughput (MB/s) (↑)						bpsp (↓)			
	Jetson AGX		DGX Spark		GPU server		8 bit		16 bit	
	Enc.	Dec.	Enc.	Dec.	Enc.	Dec.	Kodak	DIV2K	TESTIMAGES	MIT5K
Classic Methods										
PNG	6.270	110.1	12.82	226.7	7.820	172.5	4.46	4.25	13.07	13.67
JPEG-LS	40.23	56.22	92.17	138.5	63.30	81.40	4.36	4.00	10.57	11.78
WebP	6.830	102.0	16.39	216.3	14.08	206.2	3.20	3.17	–	–
JPEG2000	65.09	83.19	202.0	223.4	115.2	195.9	3.19	3.13	9.94	11.29
JPEG-XL	1.130	104.6	2.670	287.8	2.350	164.4	2.90	2.83	**9.46**	10.83
FLIF	0.360	6.070	1.000	12.90	0.952	8.878	2.90	2.91	9.72	**10.49**
IPR-L *(Ours)*	115.2 (1.8×)	122.8 (1.1×)	348.6 (1.7×)	354.4 (1.2×)	656.5 (5.7×)	826.4 (4.0×)	3.19	3.24	9.90	11.34
Learning-based Methods										
L3C	0.837	0.811	1.444	1.972	2.847	2.635	3.25	3.09	–	–
ArIB-BPS	0.061	0.060	0.0918	0.0913	0.339	0.346	**2.78**	2.55	–	–
SReC	0.910	0.903	1.853	1.730	2.725	2.524	3.04	2.94	–	–
FNLIC	0.000064	0.091	0.00012	2.517	0.001	0.315	2.88	2.75	–	–
OmniZip-S	0.124	0.207	0.377	0.344	1.020	2.125	3.31	2.97	–	–
OmniZip-M	0.032	0.075	0.144	0.113	0.483	0.781	3.10	2.75	–	–
OmniZip-L	0.010	0.027	0.059	0.045	0.208	0.275	2.93	**2.54**	–	–
FSAR	5.295	4.519	15.26	12.62	12.43	8.650	3.66	–	–	–
IPR-I *(Ours)*	0.0166 (0.003×)	29.2 (6.5×)	0.047 (0.003×)	171.5 (13.6×)	0.268 (0.022×)	515.5 (59.6×)	3.14	3.07	9.71	10.98
IPR-L *(Ours)*	**115.2** (21.8×)	**122.8** (27.2×)	**348.6** (22.8×)	**354.4** (28.1×)	**656.5** (52.8×)	**826.4** (95.5×)	3.19	3.24	9.90	11.34
+MDL	20.5 (3.9×)	101.4 (22.4×)	65.13 (4.3×)	277.2 (22.0×)	53.2 (4.3×)	405.1 (46.8×)	3.17	3.15	9.80	11.07
	Raspberry Pi 5		iPhone 13 Pro		macOS		8 bit		16 bit	
	Enc.	Dec.	Enc.	Dec.	Enc.	Dec.	Kodak	DIV2K	TESTIMAGES	MIT5K
Classic Methods										
PNG	6.286	**115.7**	8.92	**266.6**	13.51	**201.4**	4.46	4.25	13.07	13.67
JPEG-LS	**33.82**	47.97	**63.63**	75.77	83.50	99.63	4.36	4.00	10.57	11.78
WebP	4.889	109.9	10.25	125.6	14.85	188.1	3.20	3.17	–	–
JPEG2000	26.75	28.80	16.64	55.92	**119.9**	135.3	3.19	3.13	9.94	11.29
JPEG-XL	3.830	28.72	1.550	20.04	2.430	139.4	**2.90**	2.83	**9.46**	**10.83**
IPR-L *(Ours)*	24.06 (0.71×)	23.34 (0.20×)	29.11 (0.46×)	30.03 (0.11×)	77.03 (0.64×)	77.97 (0.39×)	3.19	3.24	9.90	11.34
Learning-based Methods										
OmniZip-S†	0.0029	0.0027	0.1054	0.1053	0.1488	0.1491	3.31	2.97	–	–
OmniZip-M†	0.00069	0.00065	0.0409	0.0409	0.0706	0.0706	3.10	2.75	–	–
OmniZip-L†	0.00022	0.00022	0.0133	0.0133	0.0212	0.0212	2.93	**2.54**	–	–
FSAR†	1.967	3.055	–	–	9.925	15.07	3.66	–	–	–
IPR-I *(Ours)*	0.00224 (0.0011×)	4.270 (1.4×)	0.00663 (0.063×)	8.571 (81.4×)	0.0333 (0.0034×)	24.47 (1.6×)	3.14	3.07	9.71	10.98
IPR-L *(Ours)*	24.06 (12.2×)	23.34 (7.6×)	29.11 (276×)	30.03 (285×)	77.03 (7.8×)	77.97 (5.2×)	3.19	3.24	9.90	11.34
+MDL	8.10 (4.1×)	17.74 (5.8×)	13.96 (132×)	32.10 (305×)	37.26 (3.8×)	85.53 (5.7×)	3.17	3.15	9.80	11.07

* *PILC Kang et al. (2022) — Enc/Dec 180.3/217.2 MB/s, bpsp 3.41 (DIV2K), on a Tesla V100 at 32×32. Reported, source code N/A.*

4.2 Evaluation Setup

Datasets and baselines We compare IPR with classical codecs including PNG, JPEG-LS, JPEG2000, JPEG-XL, WebP, and FLIF (Boutell, 1997; Weinberger et al., 2000; Taubman & Marcellin, 2002; Alakuijala et al., 2019; Zern et al., 2024; Sneyers & Wuille, 2016); efficient learned codecs including FSAR, SReC, L3C, and OmniZip (Zhang et al., 2024a; Cao et al., 2020; Mentzer et al., 2019; Zhao et al., 2026); and methodologically related approaches including ArIB-BPS and FNLIC (Zhang et al., 2024b; 2025). PILC (Kang et al., 2022) is included as a reported reference. We evaluate 8-bit images on Kodak (Franzen, 1999) and the DIV2K validation set (Agustsson & Timofte, 2017), and 16-bit images on TESTIMAGES (Asuni & Giachetti, 2014) and MIT5K (Bychkovsky et al., 2011). Following prior work (Punnappurath & Brown, 2021), MIT5K uses the resized HDR images numbered 4001–5000.

Table 2: **On-device lossless coding on Apple Watch SE 2.**

	Apple Watch SE 2 (256×256)		
	Throughput (MB/s) (↑)		bpsp (↓)
	Enc.	Dec.	KODAK
Apple Watch Built-in Codecs			
lzfse	**26.95**	**186.5**	6.06
zlib	12.25	97.58	5.82
lzma	1.915	15.45	4.89
PNG	0.951	20.12	3.88
Ours			
IPR-I	–	0.118	3.00
IPR-L	4.183	4.183	**2.97**
IPR-L+MDL	0.923	2.849	3.02

Table 3: **Deployment cost across learned lossless codecs.** Installed assets, parameter count, single-image encoding/decoding latency, peak memory, and FLOPs are reported for a 512×512 image. "–" denotes not applicable. ^{V} and ^{R} denote VRAM and RSS memory, respectively.

	Install (↓)		Latency (ms) (↓)		Memory (MB) (↓)				
					GPU server		macOS (CPU)		
Method	Assets	# Params	Enc.	Dec.	Enc.	Dec.	Enc.	Dec.	FLOPs (GPU) (↓)
L3C	20 MB	5.01 M	366	329	533.5^V	527.9^V	–		358.6 G
ArIB-BPS	585 MB	146.30 M	2,168	2,156	3,175.5^V	3,208.7^V	–		70.6 T
SReC	18.4 MB	4.20 M	493	340	269.4^V	204.4^V	–		229.6 G
FNLIC	281 kB	68.58 k	4,228,671	382	699.6^V	642.7^V	–		33.8 G
OmniZip-S	101 MB	4.93 M	43,970	55,566	159.9^V	159.9^V	396.6^R	392.2^R	20.9 T
OmniZip-M	599 MB	39.98 M	47,660	68,712	315.1^V	315.1^V	698.3^R	699.6^R	111.7 T
OmniZip-L	2,109 MB	159.55 M	53,000	88,180	648.6^V	648.6^V	1,504.1^R	1,506.3^R	371.8 T
FSAR	2.36 MB	587.80 k	50	79	242.0^V	122.1^V	1,957.4^R	2,167.3^R	1.58 G
IPR-I *(Ours)*	0	11,153	25,363	4.2	585.3^V	683.7^V	80.9^R	244.9^R	3.10 G
IPR-L *(Ours)*	7.0 MB	0	**3.3**	**3.4**	**10.9**^{R}	**10.6**^{R}	**25.1**^{R}	**20.9**^{R}	**0**
IPR-L+MDL *(Ours)*	0	0	68	4.3	30.0^R	19.4^R	49.8^R	49.5^R	**0**

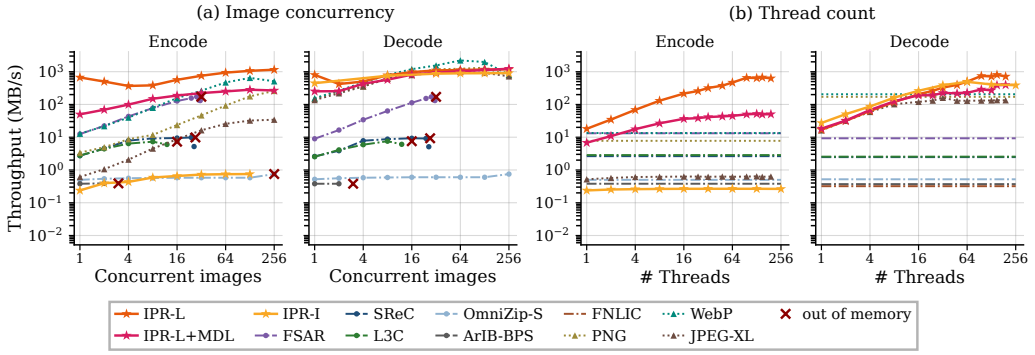


Figure 4: **Throughput scaling with concurrency and threading on our GPU server.** (a) Throughput as the number of concurrent images increases. (b) Throughput for a single image as the number of CPU threads increases. Crosses indicate out-of-memory configurations.

Hardware Table 1 evaluates coding throughput across six hardware platforms: NVIDIA Jetson AGX Orin, NVIDIA DGX Spark, a GPU server with an NVIDIA RTX PRO 6000 Max-Q, Raspberry Pi 5, iPhone 13 Pro, and MacBook Air M3. The wearable evaluation additionally uses an Apple Watch SE 2. Detailed specifications of all experimental devices are provided in Appendix Table 8.

Evaluation protocol Compression rate is reported in bits per subpixel (bpsp). Following Kang et al. (2022), throughput is measured as the uncompressed input size divided by wall-clock time. Encoding is measured end-to-end from the raw image to the compressed file, while decoding is measured from the compressed file to the reconstructed raw image. Table 1 reports single-image throughput, with one image processed per request and no inter-image concurrency. Each measurement uses one warm-up run followed by 30 timed repetitions of one image, and the table reports the mean throughput. Timings include the reversible transforms and entropy coder, and every run verifies byte-exact reconstruction. For device-side evaluation, OmniZip uses the official repository for model prediction, with the remaining coding path implemented in our framework. FSAR is ported to CPU execution for the reported measurements, but not to iOS/Xcode.

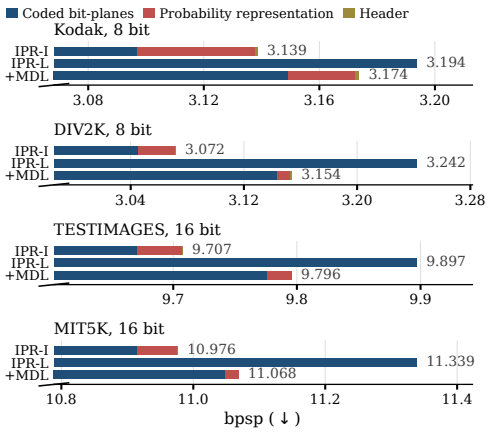


Figure 5: **Bitstream composition of IPR.** The total rate is decomposed into coded bit-planes, the representation, and header.

Table 4: **Ablation study of the IPR-I coding pipeline on Kodak.** The three blocks ablate the reversible transform, discrete finite state, and image-specific spatial code, respectively.

Transform			Kodak		*Model Inputs*				Kodak		$z = Q_{\mathcal{Z}}(\mathbf{A} \cdot \phi_f(\mathbf{k}))$				Kodak	
DWT	Bit-pl.	Color	bpsp (↓)	Δ	$\mathcal{N}(b_i)$	q	c	u	bpsp (↓)	Δ	$\mathbf{A}$	ϕ	f	$\mathcal{Z}$	bpsp (↓)	Δ
✓	✓	✓	**3.1386**	—	✓	✓	✓	✓	**3.1386**	—	✓	✓	✓	✓	**3.1386**	—
×	✓	✓	3.4695	+0.3309	×	✓	✓	✓	4.0379	+0.8993	×	✓	✓	✓	3.1875	+0.0489
✓	×	✓	3.2563	+0.1177	✓	×	✓	✓	3.3641	+0.2255	✓	×	–	✓	3.1507	+0.0121
✓	✓	×	4.4125	+1.2739	✓	✓	×	✓	3.2865	+0.1479	✓	✓	×	✓	3.1446	+0.0060
					✓	✓	✓	×	3.1795	+0.0409	✓	✓	✓	×	3.1800	+0.0414
											✓	→PE	–	✓	3.1693	+0.0307
											✓	→sin	–	✓	3.1939	+0.0553

4.3 RESULTS

Compression and throughput Table 1 highlights throughput relative to FSAR, the fastest learned method. For IPR-L, throughput relative to the fastest classical codec for each metric is also reported. IPR removes repeated neural probability inference from the coding loop, achieving competitive compression rates with substantially higher throughput. Our IPR extends directly to 16-bit images, where several learned baselines do not provide supported results. On the GPU server, IPR-I decoding and IPR-L encoding reach $59.6\times$ and $52.8\times$ the corresponding FSAR throughput, while IPR-L reaches $5.7\times/4.0\times$ the fastest classical encoding/decoding throughput. On low-core-count devices, IPR-I decoding remains 1.4–$1.6\times$ faster than FSAR where available, while IPR-L improves encoding/decoding throughput by 7.8–$12.2\times/5.2$–$7.6\times$ over FSAR on Raspberry Pi 5 and MacBook Air M3. On the iPhone 13 Pro, where FSAR is unavailable, IPR-L remains over two orders of magnitude faster than the evaluated OmniZip variants. Compared with classical codecs, however, IPR-L does not always match decoding throughput when CPU parallelism is limited. Appendix Table 9 reports the rate–throughput trade-off across codec operating points.

On-device coding Table 2 evaluates on-device lossless coding on an Apple Watch SE 2 (44 mm). For a wearable-scale workload, Kodak images are center-cropped and resized to 256×256. The comparison is restricted to codecs with working implementations on the device, including LZFSE, LZMA, zlib, and PNG, so all reported baselines reflect direct execution on the same wearable platform. IPR-I encoding is not reported because per-image fitting does not complete within a practical time budget on the watch. The lightweight execution path of IPR-L and IPR-L+MDL enables practical lossless coding on a wearable-class device. To the best of our knowledge, IPR provides the first end-to-end demonstration of learned lossless image compression on a smartwatch.

Deployment cost Table 3 quantifies deployment cost and external model dependency. Latency, memory, and FLOPs are reported for a single 512×512 image. Learned baselines require installed pretrained parameters. For example, OmniZip-L requires 2.1 GB of installed assets with 159.6 M reported parameters. IPR-I requires no installed model; its θ contains only 8.45 k parameters and is stored in the compressed file. The resulting design shifts the computational cost to per-image fitting, with 25.4 s encoding latency and 4.2 ms end-to-end decoding latency on the GPU server. IPR-L removes learned parameters by replacing neural probability evaluation with a 7.0 MB reusable LUT. MDL variant removes the installed LUT by transmitting an image-specific tree, retaining zero learned parameters and zero neural FLOPs at 68/4.3 ms encoding/decoding latency. Compared with the learned baselines, the IPR variants require substantially fewer deployment resources.

Concurrency and thread scaling In Figure 4(a), the number of concurrent images C is increased to estimate the throughput limit of each method. IPR and classical codecs share a 256-thread CPU budget across the C concurrent images. For learned baselines, C independent processes each load a model on the GPU, while OmniZip uses a batch size of $B = C$. Crosses indicate configurations exceeding GPU memory. Figure 4(b) measures single-image throughput as the number of CPU threads T increases. Learned baselines are shown at their $C = 1$ throughput because their inference path does not expose CPU thread scaling. Our IPR scales with T and exceeds the throughput of the GPU-based learned baselines using CPU execution alone. The result shows that IPR effectively uses both image-level concurrency and intra-image CPU parallelism.

Bitstream composition Figure 5 decomposes the compressed file into coded bit-planes, the transmitted probability representation, and header overhead. The decomposition suggests two observations. First, the image-specific INR representation achieves the lowest coded bit-plane rate indicating stronger probability modeling. Second, the transmitted network and MDL tree occupy only a small fraction of the compressed file. The relative overhead of the network and tree decreases as image size increases.

Ablation study Table 4 evaluates the main components of the IPR-I coding pipeline on Kodak. Among the reversible transforms, removing the color transform causes the largest degradation (3.1386 $\rightarrow$ 4.4125 bpsp), while removing DWT or bit-plane decomposition increases the rate to 3.4695 and 3.2563 bpsp. Within the finite state, the causal neighborhood contributes most, followed by the bit-plane index q and cell index c. Removing the spatial feature u increases the rate by 0.0409 bpsp, while the coefficient matrix, atom selection, and codebook provide smaller but consistent gains. Replacing the Haar representation with positional encoding from NeRF (Mildenhall et al., 2020) or sinusoidal features from SIREN (Sitzmann et al., 2020) increases the rate to 3.1693 and 3.1939 bpsp, respectively. Further ablations in Appendix Fig. 8 and Table 10 examine the effects of context width, reference-set size, and prior mixing.

5 Discussion

Lossy extension IPR extends naturally to lossy compression by omitting lower bit-planes, equivalent to coefficient quantization. We further use Gray coding (Gray, 1953) and chroma subsampling following conventional image coding practice (Wallace, 1992). Figure 6 shows the resulting rate–distortion performance. IPR remains competitive with conventional codecs at high bitrates, but degrades rapidly at low bitrates, indicating sensitivity to coarse quantization. Predicting the omitted information with the INR improves reconstruction quality up to 2.6 dB. Together with Figure 5, the result shows that the INR provides meaningful representation capacity beyond entropy modeling.

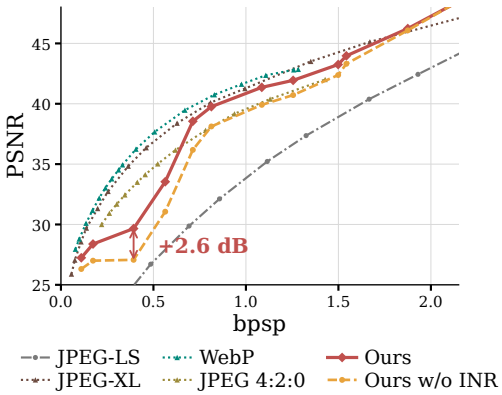


Figure 6: **Rate–distortion comparison of the lossy extension in the Kodak dataset.** PSNR (dB) is reported against bitrate in bpsp.

Deployment scenarios The IPR variants are designed for different deployment settings. A general-purpose codec remains usable on devices without a GPU, whereas many learned codecs rely on neural inference. IPR-I targets server-to-device delivery, where an image-specific probability representation is optimized once and transmitted with the compressed file. IPR-L targets repeated coding on a device, using an accumulated LUT to remove per-image optimization and neural inference. Communication across devices instead requires the probability representation to travel with the bitstream, motivating the MDL tree as a compact self-contained form of the LUT. The three variants therefore form a deployment cycle between server-side optimization, high-throughput local coding, and self-contained transmission.

Limitations IPR prioritizes deployment efficiency and throughput over compression-optimal operating points, and its rate advantage over classical codecs depends on codec settings and computational effort. IPR-I incurs per-image fitting cost, while IPR-L may lag optimized classical decoders on low-core-count devices. IPR-L also depends on a reusable probability table estimated from reference images, making performance sensitive to the calibration data.

6 Conclusion

We studied on two engineering requirements for practical learned lossless compression: decoding from a self-contained compressed file with moderate computation and deployable coding on resource-constrained devices with minimal neural inference. IPR addresses both requirements by representing the probability function in a compact deployable form. IPR-I stores an image-specific probability representation in the compressed file, IPR-L replaces repeated neural inference with a reusable LUT, and the MDL variant restores self-contained operation without an installed LUT. Experiments across heterogeneous hardware, including a smartwatch, show competitive compression with substantially reduced inference and deployment cost. The source-domain formulation also supports high-bit-depth images without changing the coding framework and extends naturally to lossy compression through bit-plane truncation. Further development of high-bit-depth and lossy coding remains a promising direction for future work.

AI USE STATEMENT

Generative AI tools were used as assistive tools for literature search, language refinement, LaTeX formatting, and feedback on manuscript organization, presentation, and consistency of notation. All AI-assisted suggestions were reviewed by the authors, and all experimental measurements, numerical results, implementations, and scientific claims were independently reviewed and validated by the authors. Generative AI was not used to generate experimental datasets or experimental measurements. The authors take full responsibility for the final content of this work.

ETHICS STATEMENT

This work studies general-purpose image compression using established public benchmark datasets and does not involve human-subject experiments, the collection of sensitive personal data, or deployment in high-risk decision-making settings.

REPRODUCIBILITY STATEMENT

Implementation and optimization details are provided in Section 4.1, while the datasets, baselines, hardware, and evaluation protocol are described in Section 4.2. The appendix further provides the mathematical notation and fixed constants, additional method derivations, the bitstream syntax, detailed device specifications, baseline configurations, and additional experiments. The supplementary material includes example compressed files produced by IPR and a viewer for decoding and inspecting the files.

REFERENCES

Eirikur Agustsson and Radu Timofte. NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

Jyrki Alakuijala, Ruud Van Asseldonk, Sami Boukortt, Martin Bruse, Iulia-Maria Comșa, Moritz Firsching, Thomas Fischbacher, Evgenii Kliuchnikov, Sebastian Gomez, Robert Obryk, et al. Jpeg xl next-generation image compression architecture and coding tools. In *Applications of digital image processing XLII*, volume 11137, pp. 112–124. SPIE, 2019.

Nicola Asuni and Andrea Giachetti. TESTIMAGES: a Large-scale Archive for Testing Visual Devices and Basic Image Processing Algorithms. In Andrea Giachetti (ed.), *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*. The Eurographics Association, 2014. ISBN 978-3-905674-72-9. doi: 10.2312/stag.20141242.

Thomas Boutell. Png (portable network graphics) specification version 1.0. Technical report, 1997.

Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. Learning photographic global tonal adjustment with a database of input / output image pairs. In *The Twenty-Fourth IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

Sheng Cao, Chao-Yuan Wu, and Philipp Krähenbühl. Lossless image compression through super-resolution. *arXiv preprint arXiv:2004.02872*, 2020.

Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser Nam Lim, and Abhinav Shrivastava. NeRV: Neural Representations for Videos. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Kecheng Chen, Pingping Zhang, Hui Liu, Jie Liu, Yibing Liu, Jiaxin Huang, Shiqi Wang, Hong Yan, and Haoliang Li. Large language models for lossless image compression: Next-pixel prediction in language space is all you need. *Advances in Neural Information Processing Systems*, 38: 157548–157569, 2025.

Gregoire Deletang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, Marcus Hutter, and Joel Veness. Language modeling is compression. In

B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 14165–14181, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/3cbf627fa24fb6cb576e04e689b9428b-Paper-Conference.pdf.

Junhao Du, Chuqin Zhou, Yunuo Chen, and Guo Lu. Large language model for lossless image compression with visual prompts. *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2026. doi: 10.1109/TCSVT.2026.3726407.

Emilien Dupont, Adam Goliński, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. Coin: Compression with implicit neural representations. *arXiv preprint arXiv:2103.03123*, 2021.

Emilien Dupont, Hrushikesh Loya, Milad Alizadeh, Adam Golinski, Yee Whye Teh, and Arnaud Doucet. COIN++: Neural compression across modalities. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=NXB0rEM2Tq.

Rich Franzen. Kodak lossless true color image suite. *source: https://r0k.us/graphics/kodak*, 1999.

Frank Gray. Pulse code communication. U.S. Patent 2,632,058, March 1953. URL https://patents.google.com/patent/US2632058A/en. Filed November 13, 1947; issued March 17, 1953.

Zongyu Guo, Gergely Flamich, Jiajun He, Zhibo Chen, and José Miguel Hernández-Lobato. Compression with bayesian implicit neural representations. *Advances in Neural Information Processing Systems*, 36:1938–1956, 2023.

Woo Kyoung Han, Byeonghun Lee, Hyunmin Cho, Sunghoon Im, and Kyong Hwan Jin. Towards lossless implicit neural representation via bit plane decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2269–2278, June 2025.

Jiajun He, Gergely Flamich, Zongyu Guo, and José Miguel Hernández-Lobato. Recombiner: Robust and enhanced compression with bayesian implicit neural representations. *arXiv preprint arXiv:2309.17182*, 2023.

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

Fatih Kamisli. Learned lossless image compression through interpolation with low complexity. *IEEE Transactions on Circuits and Systems for Video Technology*, 33(12):7832–7841, 2023. doi: 10.1109/TCSVT.2023.3273578.

Ning Kang, Shanzhao Qiu, Shifeng Zhang, Zhenguo Li, and Shu-Tao Xia. Pilc: Practical image lossless compression with an end-to-end gpu oriented neural framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3739–3748, June 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

R. Krichevsky and V. Trofimov. The performance of universal encoding. *IEEE Transactions on Information Theory*, 27(2):199–207, 1981. doi: 10.1109/TIT.1981.1056331.

Théo Ladune, Pierrick Philippe, Félix Henry, Gordon Clare, and Thomas Leguay. Cool-chic: Coordinate-based low complexity hierarchical image codec. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 13515–13522, October 2023.

Daxin Li, Yuanchao Bai, Kai Wang, Junjun Jiang, Xianming Liu, and Wen Gao. Callic: Content adaptive learning for lossless image compression. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, pp. 4679–4688, 2025.

Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Practical full resolution learned lossless image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, August 2020.

Abhijith Punnappurath and Michael S Brown. A little bit more: Bitplane-wise bit-depth recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7462–7473. Curran Associates, Inc., 2020.

Jon Sneyers and Pieter Wuille. Flif: Free lossless image format based on maniac compression. In *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 66–70, 2016. doi: 10.1109/ICIP.2016.7532320.

Yannick Strümpler, Janis Postels, Ren Yang, Luc Van Gool, and Federico Tombari. Implicit neural representations for image compression. In *European Conference on Computer Vision*, pp. 74–91. Springer, 2022.

Kaiyuan Tang, Daniel Burke, and Chaoli Wang. Lossless-inr: Lossless volumetric implicit neural representations. *arXiv preprint arXiv:2607.18150*, 2026.

David S. Taubman and Michael W. Marcellin. *JPEG2000 - image compression fundamentals, standards and practice*, volume 642 of *The Kluwer international series in engineering and computer science*. Kluwer, 2002. ISBN 978-0-7923-7519-7. doi: 10.1007/978-1-4615-0799-4. URL https://doi.org/10.1007/978-1-4615-0799-4.

Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016a. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/b1301141feffabac455e1f90a7de2054-Paper.pdf.

Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1747–1756, New York, New York, USA, 20–22 Jun 2016b. PMLR. URL https://proceedings.mlr.press/v48/oord16.html.

G.K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, 1992. doi: 10.1109/30.125072.

Peng-Shuai Wang, Yang Liu, Yu-Qi Yang, and Xin Tong. Spline Positional Encoding for Learning 3D Implicit Signed Distance Fields. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 1091–1097. International Joint Conferences on Artificial Intelligence Organization, 8 2021. Main Track.

Marcelo J Weinberger, Gadiel Seroussi, and Guillermo Sapiro. The loco-i lossless image compression algorithm: Principles and standardization into jpeg-ls. *IEEE Transactions on Image processing*, 9(8):1309–1324, 2000.

J. Zern, P. Massimino, and J. Alakuijala. WebP Image Format. RFC 9649, November 2024. URL https://www.rfc-editor.org/info/rfc9649/.

Yufeng Zhang, Hang Yu, Jianguo Li, and Weiyao Lin. Finite-state autoregressive entropy coding for efficient learned lossless compression. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 45616–45640, 2024a. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/c7138635035501eb71b0adf6ddc319d6-Paper-Conference.pdf.

Zhe Zhang, Huairui Wang, Zhenzhong Chen, and Shan Liu. Learned lossless image compression based on bit plane slicing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27579–27588, June 2024b.

Zhe Zhang, Zhenzhong Chen, and Shan Liu. Fitted neural lossless image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23249–23258, June 2025.

Yan Zhao, Zhengxue Cheng, Junxuan Zhang, Dajiang Zhou, Qunshan Gu, Qi Wang, and Li Song. Omnizip: Learning a unified and lightweight lossless compressor for multi-modal data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5337–5347, June 2026.

A APPENDIX

The appendix summarizes the notation, expands the method derivations, specifies the bitstream syntax, and provides additional experimental settings and results. The supplementary material includes example compressed files produced by IPR and a viewer for decoding and displaying the compressed files.

A.1 NOTATION

Tables 5 and 6 separate mathematical notation from serialized field names and fixed implementation constants. The tables summarize the notation used in the main text and supplementary material, respectively, for ease of reference. The main text suppresses the patch index for $\mathbf{A}$ and f. In the syntax tables, j indexes per-patch atom indices, coefficients, and scale exponents.

Table 5: **Notation of the main text** — elements, sets, and functions.

Symbol	Definition	Description	Example / Meaning / Note
Elements			
X	image	the source	one $W \times H \times 3$ image
$\mathbf{x}, x_i, N$	$\mathbf{x} = (x_1, \ldots, x_N)$	source symbols and their count	$x_i \in \{0, \ldots, 255\}$ at 8 bit
$\mathbf{b}, b_i, M$	$\mathbf{b} = \mathcal{R}(X) = (b_1, \ldots, b_M), b_i \in \{0, 1\}$	binary coding symbols and their count	M is the bit-plane sequence length
q_i	$\in \{0, \ldots, \texttt{nplanes}-1\}$	bit-plane index of b_i	11 in the 8-bit dense-domain calculation; 18 at 16 bit
c_i	$\in \{0, \ldots, 38\}$	cell index: colour channel × subband	$3 \times 13 = 39$ cells
s_i	$:= (\mathcal{N}(b_i), q_i, c_i)$	shared discrete coding state	Eq. 5; shared by IPR-I, IPR-L, and its MDL specialization
$\mathbf{k}_i$	$\in \mathbb{R}^2$	spatial coordinate of b_i	input of the Haar encoding
$\mathbf{A}$	$\in \mathbb{R}^{4\times 32}$ per patch	image-specific coefficient matrix	transmitted quantised, Eq. 7
f	subset of the Haar dictionary	selected atoms	32 of 512, per patch
u_i	$= \mathbf{A}\phi_f(\mathbf{k}_i) \in \mathbb{R}^4$	continuous spatial code	Eq. 7
z_i	$= Q_{\mathcal{Z}}(u_i) \in \mathcal{Z}$	quantised spatial code	Eq. 9
θ, θ_X	trainable parameters	of p_θ; θ_X is the per-image head of IPR-I	two ReLU hidden layers; excludes the spatial representation
λ	$\in \mathbb{R}_{\geq 0}$	weight of $\|\mathbf{A}\|_1$ in Eq. 8	10^{-4}, first half of training
$n_s, n_s^{(1)}$	$\in \mathbb{Z}_{\geq 0}$	coding events at state s, and those with $b = 1$	pooled over reference images, Eq. 13
ℓ	leaf of the MDL tree	a group of states sharing one probability	Eq. 14
$S_\ell^{(v)}$	$\in \mathbb{Z}_{\geq 0}$	current-image symbols of value v in leaf ℓ	
α	$\in \mathbb{R}_{\geq 0}$	prior strength in IPR-L estimation and MDL leaf estimation	32 for IPR-L; 8 for IPR-L+MDL
$\Delta, \Delta L_{\text{tree}}$	$\in \mathbb{R}$	coding-length gain of a split, and the split's description cost	Eq. 15; 9-bit node, 17-bit leaf
Sets			
$x_{<i}$	$= (x_1, \ldots, x_{i-1})$	full causal context	Eq. 1
$\mathcal{C}_i$	restricted causal context	what b_i is conditioned on	$= s_i$ for IPR-L, $= (s_i, z_i)$ for IPR-I
$\mathcal{N}(b_i)$	$\in \{0, 1\}^{13}$	binary causal neighbourhood of b_i	3 own upper bits, 6 neighbour features, 4 current-plane bits
$\mathcal{S}$	state space of s_i	finite by construction	$\|\mathcal{S}\| = \texttt{nplanes} \cdot 39 \cdot 2^{13}$
$\mathcal{Z}$	codebook, $\|\mathcal{Z}\| = 32$	k-means centroids of the u_i of one image	transmitted, Eq. 9
Functions			
$p_\theta(\cdot)$	$\mathbf{x} \mapsto p_\theta(\mathbf{x})$	autoregressive source model	Eq. 1
$\mathcal{R}, \mathcal{R}^{-1}$	$X \mapsto \mathbf{b}$	YCoCg-R, integer CDF 5/3, bit-plane decomposition	exactly invertible, Eq. 2
$\pi_\theta(\cdot)$	$\mathcal{C}_i \mapsto p_\theta(b_i{=}1 \mid \mathcal{C}_i)$	Bernoulli probability of a coding symbol	Eq. 3
$\pi_{\theta_X}(s, z)$	IPR-I's π_θ	per-image head over state and spatial code	Eq. 6
$\widetilde{\pi}_I(s)$	$= \mathbb{E}_{X,z\mid s}[\pi_{\theta_X}(s, z)]$	model-derived reusable prior	Eq. 12
$\pi_L(s)$	$= \dfrac{n_s^{(1)} + \alpha\widetilde{\pi}_I(s) + \frac{1}{2}}{n_s + \alpha + 1}$	pooled counts smoothed with the INR prior and KT pseudocounts	Eq. 13
π_ℓ	Eq. 14	leaf probability of the MDL tree	
$\mathcal{T}_b[\cdot], \mathcal{T}_X, \mathcal{T}_L$	$\mathcal{C}_i \mapsto \pi(\mathcal{C}_i)$	the lookup tables: generic, per-image, reusable	$\|\mathcal{S}\|\|\mathcal{Z}\|$ and $\|\mathcal{S}\|$ entries
$\phi_f(\cdot)$	$\mathbf{k} \mapsto \mathbb{R}^{32}$	Haar basis functions selected by f	fixed dictionary, Eq. 7
$Q_{\mathcal{Z}}(\cdot)$	$u \mapsto \arg\min_{z\in\mathcal{Z}} \|u - z\|$	nearest-codeword quantiser	Eq. 9
$\text{BCE}(\cdot, \cdot)$	mean binary cross-entropy	Bernoulli fitting loss; ideal rate up to averaging and log base	Eq. 8
$L_{\text{data}}(\ell; \pi)$	$-S_\ell^{(1)} \log_2 \pi - S_\ell^{(0)} \log_2(1-\pi)$	ideal coding length of a leaf	Eq. 15

Table 6: **Notation of the supplement** — the serialized names of Table 7.

Symbol	Definition	Description	Example / Meaning / Note
Elements (scalars in headers, or normative constants)			
`width`, `height`	$\in \mathbb{N}$	decoded image size before padding	
`nplanes`	$\in \mathbb{N}$	number of coded bit-planes	range of q_i; 11 in the dense-domain calculation
`coeff_lo`	$\in \mathbb{Z}$	transformed-coefficient lower bound	coded offset $\in [0, 2^{\texttt{nplanes}})$
`pside`	$= 128$	patch side	normative
`npatch`	$\in \mathbb{N}$	number of patches, $\lceil \texttt{width/pside} \rceil \cdot \lceil \texttt{height/pside} \rceil$	24 for a Kodak image
`nlevels`	$= 4$	DWT levels	LL is 8×8
`ncells`	$= 39$	channel–subband cells	range of c_i
`nsubbands`	$= 13$	subbands per channel	
`nctx`	$= 13$	causal-context width in bits	$\lvert\mathcal{N}(b_i)\rvert$
`nown`, `nnbr`, `nsameplane`	$(3, 3, 4)$ for the raster profile	own upper bits, upper-plane neighbours, current-plane neighbours	$3 + 2 \times 3 + 4 = 13$
`hdim`, `indim`	$\in \mathbb{N}$	hidden and input width of the head	hidden width 32 on Kodak, 64 elsewhere; input width is stored in the model header
`ncode`	$= 32$	codebook entries	$\lvert\mathcal{Z}\rvert$
`sdim`	$= 4$	spatial-code dimension	$\dim u_i$
`nsel`, `ndict`	$= 32, 512$	atoms selected per patch, dictionary size	$\lvert f\rvert$ and the pool f is drawn from
`nnode`	$\in \mathbb{N}$	MDL-tree nodes, leaves included	$\approx 2{,}000$ on Kodak
`k`	$\in \{0, \ldots, 15\}$	lowest planes omitted, in `flags`	0 = lossless
`f0`	$\in [1, 2^{16}-1]$	a table entry: $P(b{=}0) \cdot 2^{16}$, clipped	$= \mathrm{clip}\big((1-\pi)2^{16}\big)$
Sets (serialised arrays; j indexes patches)			
`w1`, `w2`, `w3`	$s(12)$ storage slots	W8-quantized head weights stored in signed 12-bit slots	see Section A.3
`w1_exp`, `w2_exp`, `w3_exp`	$s(8)$ per row	power-of-two scale of each weight row	
`b1`, `b2`, `b3`	$s(16)$ per row	head biases	
`codebook`	`ncode` $\times$ `sdim` $\times$ $s(16)$, `q10`	the centroids	$\mathcal{Z}$
`atom_idx[j]`	`nsel` $\times$ $u(9)$	selected Haar atoms of patch j	f
`a_q[j]`, `a_exp[j]`	`sdim` $\times$ `nsel` $\times$ $s(4)$, $s(8)$	quantised coefficients and their scale	$\mathbf{A} = \texttt{a_q[j]} \cdot 2^{\texttt{a_exp[j]}}$
`tree`	`nnode` nodes, MSB-first	the MDL tree, preorder	expands to $\mathcal{T}_L$ specialised to the image
`payload[j]`	byte[`payload_len[j]`]	entropy-coded bit-planes of patch j	binary coding symbols of patch j
Functions (descriptors and grammars)			
$u(n)$, $s(n)$	n-bit integer	unsigned; two's-complement signed	LSB-first in packed sections
`q10`	fixed point	signed, 10 fractional bits	codebook entries
`node()`	Fig. 7	recursive tree grammar	leaf $= u(1)\, u(16)$, internal $= u(1)\, u(4)\, u(4)$
`encode_bit()`	Fig. 7	one range-coder step on $(\texttt{f0}, b_i)$	byte-renormalised, no floating point

A.2 Additional Method Details and Derivations

We expand the finite-state, reusable-probability, and MDL formulations of Section 3 with derivations omitted from the main text for space. The coding procedure remains unchanged.

A.2.1 Finite state space and LUT size

The shared state $s_i = (\mathcal{N}(b_i), q_i, c_i)$ in Eq. 5 combines a binary context, a bit-plane index, and a channel–subband index. The dense lookup domain includes every combination of the three components, including states absent from an individual image. Its cardinality is

$$|\mathcal{S}| = 2^{\texttt{nctx}} \cdot \texttt{nplanes} \cdot \texttt{ncells}. \tag{16}$$

Using `nctx`$=13$, `ncells`$=39$, and `nplanes`$=11$ for the reported 8-bit dense domain gives

$$|\mathcal{S}| = 2^{13} \times 11 \times 39 = 3{,}514{,}368. \tag{17}$$

By comparison, even a pixel-domain predictor conditioned on only four 8-bit RGB pixels has $2^{4\times3\times8} = 2^{96}$ possible contexts. Storing a 256-way PMF for each context requires 2^{104} probability entries, corresponding to approximately 4.06×10^{19} TB with 16-bit probabilities. Direct LUT materialization is therefore infeasible even for such a small pixel neighborhood. The bit-plane formulation avoids this combinatorial explosion through a compact finite binary state.

IPR-L stores one 16-bit probability per state, so the dense table requires

$$\begin{aligned} B_{\text{IPR-L}} &= 16|\mathcal{S}| = 56{,}229{,}888 \text{ bits} \\ &= 7{,}028{,}736 \text{ bytes} \approx 7.03 \text{ MB}. \end{aligned} \tag{18}$$

IPR-I additionally indexes the spatial code $z \in \mathcal{Z}$ in Eq. (10). With $|\mathcal{Z}| = \texttt{ncode} = 32$, its dense table requires

$$\begin{aligned} B_{\text{IPR-I}} &= 16|\mathcal{S}||\mathcal{Z}| = 1{,}799{,}356{,}416 \text{ bits} \\ &= 224{,}919{,}552 \text{ bytes} \approx 224.9 \text{ MB}. \end{aligned} \tag{19}$$

The quoted sizes use decimal MB, with $1\ \text{MB} = 10^6$ bytes, matching the main text. The 224.9 MB value denotes a fully materialized dense IPR-I table. The portable implementation evaluates entries lazily and caches only those requested during coding. The compressed file carries the compact probability representation required to reconstruct the lookup function, rather than the dense table. The GPU-server IPR-I implementation evaluates required entries with CUDA during decoding. IPR-L retains the finite-state lookup principle while removing the spatial-code axis.

A.2.2 Bernoulli coding length and representation cost

Taking the negative base-two logarithm of the Bernoulli factors in Eq. 3 gives the ideal contribution of an observed bit under IPR-I,

$$\ell_i = -b_i \log_2 \pi_{\theta_X}(s_i, z_i) - (1 - b_i) \log_2(1 - \pi_{\theta_X}(s_i, z_i)) \,. \tag{20}$$

The ideal data description length is therefore

$$L_{\text{data}}^{\text{IPR-I}} = \sum_{i=1}^{M} \ell_i. \tag{21}$$

For fixed probability predictions, the BCE term in Eq. 8 optimizes the same Bernoulli log-loss up to averaging and the logarithm convention. Fitting uses the continuous Haar features, whereas entropy coding uses the quantised representation compiled into the LUT of Eq. (10). The ℓ_1 penalty regularizes the spatial coefficients rather than measuring their exact serialized length. Parameter quantization, integer probability precision, range-coder termination, and byte alignment also separate the floating-point fitting loss from the final file length. The conceptual total-rate decomposition is

$$\begin{aligned} L_{\text{total}}^{\text{IPR-I}} \approx\ & L_{\text{data}}^{\text{IPR-I}} + L(\theta_X) + L(\mathbf{A}) \\ & + L(f) + L(\mathcal{Z}) + L_{\text{header}}. \end{aligned} \tag{22}$$

Here $L(\theta_X)$, $L(\mathbf{A})$, $L(f)$, and $L(\mathcal{Z})$ describe the transmitted probability representation, with the costs of $\mathbf{A}$ and f summed over all patches. The decomposition corresponds to the coded bit-planes, probability representation, and header in the bitstream-composition analysis.

Representation entry count. Table 3 counts the full image-specific representation rather than the probability head alone. For the reported 512×512, $Q = 9$ benchmark, the 11,153 entries comprise 8,449 head parameters, 2,048 coefficients in $\mathbf{A}$, 512 selected Haar indices, 16 scale/exponent entries, and 128 codebook entries. The last four counts correspond to 16 patches and a 32×4 codebook. The entry count differs from the serialized byte size because the components use different storage widths.

A.2.3 From marginalization to pooled IPR-L estimation

Equation 11 defines the target marginal,

$$p_L(b = 1 \mid s) = \sum_X \sum_z p(b = 1 \mid s, z, X)\, p(z, X \mid s). \tag{23}$$

The image-specific models define the reusable marginal prior in Eq. 12,

$$\widetilde{\pi}_I(s) = \sum_X \sum_z \pi_{\theta_X}(s, z)\, p(z, X \mid s). \tag{24}$$

Let $\mathcal{D}_{\text{ref}} = \{(X_j, z_j, s_j, b_j)\}_{j=1}^{M_{\text{ref}}}$ be the indexed collection of reference coding events, including repetitions. The pooled counts are

$$n_s = \sum_{j=1}^{M_{\text{ref}}} \mathbf{1}[s_j = s], \tag{25}$$

$$n_s^{(1)} = \sum_{j=1}^{M_{\text{ref}}} \mathbf{1}[s_j = s,\ b_j = 1]. \tag{26}$$

For $n_s > 0$, the implementation estimates $p(z, X \mid s)$ by the fraction of reference events with the corresponding (X, z, s). The resulting prior is the event-weighted average

$$\widetilde{\pi}_I(s) = \frac{1}{n_s} \sum_{j:s_j=s} \pi_{\theta_{X_j}}(s, z_j). \tag{27}$$

Probabilities are averaged without rounding them to binary decisions. For $n_s = 0$, we evaluate $\pi_{\theta_X}(s, z)$ for each of the eight reference-image models and all 32 codewords of each model, and use their uniform average as $\widetilde{\pi}_I(s)$. This defines a uniform backoff for $p(z, X \mid s)$ when no reference event is available at s; the same prior-and-KT smoothing rule is then applied. Combining the prior, pooled counts, and KT half-counts gives Eq. 13,

$$\pi_L(s) = \frac{n_s^{(1)} + \alpha\widetilde{\pi}_I(s) + \frac{1}{2}}{n_s + \alpha + 1}. \tag{28}$$

The prior strength is $\alpha = 32$ for IPR-L. Empirical counts dominate when $n_s \gg \alpha + 1$; the marginal prior and KT half-counts dominate at sparse states. For $n_s = 0$, the estimator becomes $(\alpha\widetilde{\pi}_I(s) + 1/2)/(\alpha + 1)$. The same first eight images provide the fitted models and the pooled counts. Model fitting and LUT construction occur offline; subsequent image coding uses the fixed LUT without per-image fitting. Setting $\alpha = 0$ recovers count-based KT; the prior-only variant uses $\widetilde{\pi}_I(s)$ directly. Table 10 compares both variants with the hybrid estimator.

A.2.4 MDL TREE OBJECTIVE AND SPLIT COST

Let $\mathcal{T}$ denote the transmitted tree and $\mathcal{L}(\mathcal{T})$ its leaves. Writing $L_{\text{tree}}(\mathcal{T})$ for the serialized tree-body length, the MDL objective is

$$L_{\text{MDL}}(\mathcal{T}) = L_{\text{tree}}(\mathcal{T}) + \sum_{\ell \in \mathcal{L}(\mathcal{T})} L_{\text{data}}(\ell; \pi_\ell), \tag{29}$$

where the ideal leaf coding length follows Eq. 15,

$$L_{\text{data}}(\ell; \pi) = -S_\ell^{(1)} \log_2 \pi - S_\ell^{(0)} \log_2(1 - \pi). \tag{30}$$

The usual convention $0 \log_2 0 = 0$ applies to zero-count terms. Replacing a leaf ℓ by an internal node and two leaves ℓ_L, ℓ_R gives

$$\begin{aligned} \Delta = L_{\text{data}}(\ell; \pi_\ell) - L_{\text{data}}(\ell_L; \pi_{\ell_L}) \\ - L_{\text{data}}(\ell_R; \pi_{\ell_R}) - \Delta L_{\text{tree}}. \end{aligned} \tag{31}$$

The objective after the split equals the objective before the split minus Δ. Hence $\Delta > 0$ is exactly the condition for reducing the stated description-length objective, recovering the split rule in Eq. 15. A leaf contains a one-bit tag and a 16-bit probability, for $1 + 16 = 17$ bits. An internal node contains a one-bit tag, a four-bit feature index, and a four-bit threshold, for $1 + 4 + 4 = 9$ bits. Replacing one leaf by one internal node and two leaves therefore costs

$$\Delta L_{\text{tree}} = 9 + 2 \cdot 17 - 17 = 26 \text{ bits}. \tag{32}$$

The 26-bit increment is the tree-body penalty used by the split decision. Fixed container fields and final byte padding are excluded from the per-split increment. The MDL objective combines the exact tree-body cost with ideal data lengths, rather than equating the split gain with an exact change in complete file size.

A.2.5 INTERPRETATION OF THE PRIOR-REGULARIZED LEAF PROBABILITY

Let $\mathcal{O}_\ell$ be the occupied states assigned to leaf ℓ, and let $K_\ell = |\mathcal{O}_\ell|$. The reusable probability is averaged uniformly over occupied states,

$$\bar{\pi}_{L,\ell} = \frac{1}{K_\ell} \sum_{s \in \mathcal{O}_\ell} \pi_L(s). \tag{33}$$

Equation 14 combines current-image counts with this prior and KT half-counts, using $\alpha = 8$ per leaf:

$$\pi_\ell = \frac{S_\ell^{(1)} + \alpha\bar{\pi}_{L,\ell} + \frac{1}{2}}{S_\ell^{(1)} + S_\ell^{(0)} + \alpha + 1}. \tag{34}$$

For $N_\ell = S_\ell^{(1)} + S_\ell^{(0)} > 0$, define $\widehat{p}_\ell = S_\ell^{(1)}/N_\ell$. Then

$$\pi_\ell = \frac{N_\ell}{N_\ell + \alpha + 1}\widehat{p}_\ell + \frac{\alpha}{N_\ell + \alpha + 1}\bar{\pi}_{L,\ell} + \frac{1}{N_\ell + \alpha + 1}\frac{1}{2}. \tag{35}$$

The current-image counts dominate when $N_\ell \gg \alpha + 1$. The decoder receives the quantized leaf probabilities and does not recompute the prior average.

IPR-I transmits a compact parameterization of the finite probability function and evaluates lookup entries lazily. IPR-L stores the marginalized function as a reusable LUT. The MDL specialization of IPR-L instead transmits an image-specific partition of the shared state space and leaf probabilities regularized by the reusable IPR-L statistics, removing the need for a provisioned LUT at the decoder.

A.3 Bitstream Syntax

The deployment forms share the transformed source representation and range-coded payload grammar, but differ in how the decoder obtains the probability table. IPR-I carries an image-specific probability representation, IPR-L uses a provisioned reusable table, and the IPR-L+MDL specialization carries a tree that expands over the reusable state space. Table 7 summarizes the serialized fields.

Conventions. The descriptors $u(n)$ and $s(n)$ denote unsigned and signed n-bit integers, respectively. Signed integers use two's complement. Header integers are little-endian and byte-aligned. Packed model arrays and patch codes are LSB-first, with zero padding only at section ends. The preorder tree is MSB-first and zero-padded in the final byte. All sections start at byte boundaries. The `q10` format has 10 fractional bits. The writer rejects values outside the specified field range. Typewriter names denote serialized fields; Section A.1 gives their mathematical counterparts.

Fixed coding profile. Images are reflect-padded to multiples of `pside`$= 128$ and partitioned into raster-ordered patches, giving $\texttt{npatch} = \lceil \texttt{width}/128 \rceil \lceil \texttt{height}/128 \rceil$. YCoCg-R and four levels of integer CDF 5/3 lifting produce 13 subbands per channel and 39 channel–subband cells. The raster profile stores $(\texttt{nctx}, \texttt{nown}, \texttt{nnbr}, \texttt{nsameplane}) = (13, 3, 3, 4)$ and uses the `LNM2` model tag. Let $v(y, x)$ be the nonnegative transformed-coefficient offset and q the current bit-plane, with $q = 0$ the least significant plane. Context bits 0–2 are the current coefficient's bits $q + 1$, $q + 2$, and $q + 3$. Bits 3–8 contain two features from each of N, S, and W, in that order: the bit at $q + 1$ and the indicator $\mathbf{1}[v \gg (q + 2) \neq 0]$. The indicator is the OR of all bits above $q + 1$, not a single upper-plane bit. Bits 9–12 are the current-plane bits at W, N, NW, and NE. Out-of-patch neighbours contribute zero. Planes are decoded from most to least significant, with row-major traversal within each channel and plane. The S feature is causal because it reads only previously decoded upper planes. The encoder and decoder produced identical context traces for all 491,520 events of the verified Kodak patch. The older `LNM1` profile has $(13, 3, 5, 0)$: three own upper bits and two upper-plane features from each of N, S, W, E, and NW.

IPR-I uses 32 selected Haar atoms from a dictionary of 512, a four-dimensional spatial code, and a 32-entry codebook. Model and tree headers carry the corresponding profile metadata where specified. The fields `nplanes` and `coeff_lo` define the coefficient interval $[\texttt{coeff_lo}, \texttt{coeff_lo} + 2^{\texttt{nplanes}})$. The bit-plane index ranges from zero to `nplanes` $- 1$; 11 planes are used in the 8-bit dense-domain calculation and 18 in the reported 16-bit configuration. No separate source-bit-depth field is stored. IPR-L obtains the compatible state geometry and coefficient range from the provisioned table.

Selective LUT baking. The optional `bake_indices` field in Table 7 lets the encoder transmit a cover of the LUT states visited by the image. For each bit plane and channel–subband cell, the encoder collects the context indices used during coding and retains every aligned block of $B = 2^r$ consecutive indices that contains a visited index ($B = 4$ in the B4 configuration). Thus, a block with index g covers $[gB, (g + 1)B)$; the cover may include unused entries but contains every required context. The sorted, unique block identifiers, flattened over plane, cell, and context block, are stored as gap-coded variable-length integers compressed with Zstd, preceded by a 16-byte `ICOV` descriptor specifying the block shift r and stream lengths. The decoder expands this cover into block offsets, reconstructs the spatial codes from the transmitted model and patch codes, and batches GPU

```
node() {                          // preorder, MSB-first
    is_leaf    u(1)
    if (is_leaf) {
        f0     u(16)              // P(bit = 0) at 16-bit precision
    } else {
        feature    u(4)           // 0: plane, 1: subband, 2: channel,
        threshold  u(4)           // 3+i: context bit i, i = 0,...,12
        node()                    // left:  feature < threshold
        node()                    // right: feature >= threshold
    }
}

encode_bit(f0, bit) {             // low: uint64; range: uint32
    bound = (range >> 16) * f0    // initial range = 2^32 - 1
    if (bit == 0) range = bound
    else { low += bound; range -= bound; }
    while (range < 2^24) { range <<= 8; shift_low(); }
}                                 // one dummy byte; five-byte flush
```

Figure 7: **Tree syntax and range-coder step.** The 9-bit internal node and 17-bit leaf yield the 26-bit split penalty in Eq. 32. The coder uses integer arithmetic and independent patch streams. Equal bit and probability sequences therefore produce equal coded bytes across devices.

evaluation over covered contexts and the codewords used in each cell, skipping structurally impossible contexts. The CPU entropy decoder then forms each causal context and indexes the resulting compact probability buffer by its block offset, within-block index, and spatial code. Compared with dense baking, the primary aim is lower decoding latency through fewer head evaluations and transfers, with reduced probability-buffer storage as an additional benefit; the cover metadata adds a small bitstream overhead.

Container order. An IPR-L stream begins with `LNRC`, followed by the width and height and then the common patch records. The bare stream carries neither a version field nor a LUT identifier and must be interpreted against the provisioned table. The IPR-L+MDL container begins with `LNRA` and `tree_len`, followed by the `LNRT` blob and a complete `LNRC` stream. The IPR-I v2 header contains `IPRI`, version, width, height, the three section lengths, flags, and the 32-byte digest slot, totaling 64 bytes. The model, patch-code, and payload sections follow in that order. The IPR-I payload contains patch records directly, without an additional `LNRC` prefix or repeated dimensions. Flag bit 0 enables digest verification; the digest slot remains part of the fixed header. The v2 representation regenerates the fixed sigmoid and Haar tables from codec constants rather than transmitting them. Head weights are W8-quantized and stored in signed 12-bit slots. Each patch code occupies $32 \cdot 9 + 8 + 4 \cdot 32 \cdot 4 = 808$ bits, or 101 bytes. For the supplementary demo, the self-contained IPR-L+MDL stream is wrapped by a 64-byte `IPRT` header and stored with the `.iprt` extension; the wrapped payload is the `LNRA` container described below.

Tree and payload decoding. The `LNRT` v2 header occupies 52 bytes and precedes the tree grammar in Fig. 7. Each leaf supplies one 16-bit probability to all states assigned to the leaf. The receiver expands the tree into a lookup table and uses the IPR-L coding kernel without the provisioned LUT. For lossless coding, each patch is processed from the highest to the lowest plane, then over the Y, Co, and Cg channels and raster-ordered coefficients. The byte-coder containers are `IPRC`, `LNRC`, and `LNRA`. The range coder uses `f0` $\in \{1, \ldots, 65535\}$ for the probability of zero, a 64-bit lower-bound accumulator, a 32-bit range, and byte renormalization. Each patch stream is preceded by a 32-bit byte length, supporting independent patch decoding once the probability representation is available. The decoder applies inverse CDF 5/3 and inverse YCoCg-R transforms and removes padding to recover the original image.

A.4 Experimental Setup and Devices

Table 8 lists the systems used for the main throughput comparisons and wearable evaluation.

The MacBook Air and Raspberry Pi implementations use `-O3`, with Apple clang 21 and GCC 14.2, respectively. The iPhone and Apple Watch builds use the Xcode Release configuration with the platform-default `-Os` optimization. On the MacBook Air M3, the two optimization levels differ by approximately 2% for IPR. The Raspberry Pi 5 is evaluated without active cooling, with the CPU

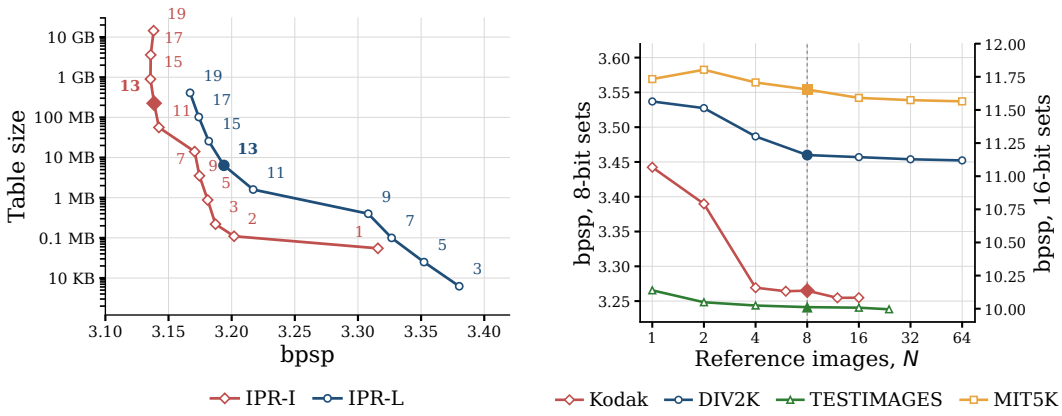
Table 7: **Serialized fields of the deployment containers.** Fields are grouped by function; container order is specified in Section A.3. `IPRI`, `LNRC`, and `LNRA` identify IPR-I, IPR-L, and the IPR-L+MDL specialization, respectively. The IPR-L table is provisioned externally and is absent from the stream. j indexes patches in raster order.

Syntax element	Semantics	Coding	Condition
Container fields			
`magic`	`IPRI` (`IPRC` for range-coded IPR-I), `LNRC`, or `LNRA`; selects the deployment form	byte[4]	each container
`version`	2; 3 with optional bake indices	$u(32)$	`IPRI`/`IPRC`
`width`, `height`	image dimensions before padding; `LNRA` obtains both from its embedded `LNRC` stream	$2 \times u(32)$	`IPRI`, `LNRC`
`tree_len`	byte length of the following `LNRT` blob	$u(32)$	`LNRA`
`model_len`, `codes_len`, `payload_len`	byte lengths of the model, patch codes, and payload sections	$3 \times u(32)$	`IPRI`
`flags`	bit 0: verify stored digest; bit 1: legacy enclosed constants; bits 8–11: omitted lowest planes k, with $k = 0$ for lossless coding	$u(32)$	`IPRI`
`sha256`	fixed digest slot for decoded RGB; checked when flags bit 0 is set	byte[32]	`IPRI`
`bake_indices` (optional)	LUT context-block cover; after the 64-byte header, before the model	16-byte `ICOV` descriptor + Zstd-compressed gap varints	`IPRC` v3; other fields as `IPRI`
Probability-model header and arrays			
`magic`, `version`	model tag `LNM1` (`nsameplane` $= 0$) or `LNM2` (`nsameplane` > 0, carrying one additional profile field), or tree tag `LNRT`; section version	byte[4], $u(32)$	`IPRI`, `LNRA`
`nplanes`, `coeff_lo`	coefficient offset lies in $[0, 2^{\texttt{nplanes}})$ after subtracting `coeff_lo`	$u(32)$, $s(32)$	model/tree header
`hdim`, `indim`, `npatch`	head hidden/input widths and number of patches	$3 \times u(32)$	model header
`nnode`	tree-node count, including leaves	$u(32)$	tree header
profile fields	context: `nctx`, `nown`, `nnbr`, `nsameplane`; geometry: `ncells`, `pside`, `nlevels`, `nsubbands`; spatial code: `ncode`, `sdim`, `nsel`, `ndict`	$u(32)$ per present field	model/tree header; profile-dependent subset
`w1`	first-layer W8-quantized weights stored in signed 12-bit slots, scaled row-wise by $2^{\texttt{w1_exp}}$	`hdim` $\times$ `indim` $\times\ s(12)$	`IPRI`
`w1_exp`, `b1`	row-wise scale exponent and bias	`hdim` $\times\ s(8)$; `hdim` $\times\ s(16)$	`IPRI`
`w2`, `w2_exp`, `b2`	second-layer W8-quantized weights stored in signed 12-bit slots; row-wise exponents and biases	`hdim`$^2 \times\ s(12)$; `hdim` $\times\ s(8)$; `hdim` $\times\ s(16)$	`IPRI`
`w3`, `w3_exp`, `b3`	output-layer W8-quantized weights stored in signed 12-bit slots; exponent and bias	`hdim` $\times\ s(12)$; $s(8)$; $s(16)$	`IPRI`
`codebook`	spatial-code centroids in `q10`	`ncode` $\times$ `sdim` $\times\ s(16)$	`IPRI`
`tree`	preorder tree; grammar in Fig. 7	9-bit internal node; 17-bit leaf	`LNRA`, `nnode` nodes
Image-specific codes, per patch j			
`atom_idx[j]`	selected Haar dictionary indices	`nsel` $\times\ u(9)$	`IPRI`
`a_exp[j]`	power-of-two step exponent	$s(8)$	`IPRI`
`a_q[j]`	coefficient matrix: $\mathbf{A} =$ `a_q[j]` $2^{\texttt{a_exp[j]}}$	`sdim` $\times$ `nsel` $\times\ s(4)$	`IPRI`
Common payload records, per patch j			
`payload_len[j]`	byte length of the following patch stream	$u(32)$	all forms
`payload[j]`	independently entropy-coded bit-planes	byte[`payload_len[j]`]	all forms

Table 8: **Detailed specifications of the experimental devices.** Cores denote physical CPU cores.

Device	SoC / CPU	Cores	Memory	OS	Accelerator
GPU server	2× AMD EPYC 9354 (Zen 4, 32C each, up to 3.8 GHz)	64	768 GB	Ubuntu 22.04.3 LTS (Linux 5.15)	NVIDIA RTX PRO 6000 Blackwell Max-Q, 96 GB (1 of 8 used)
DGX Spark	NVIDIA GB10, 10×X925 + 10×A725	20	128 GB	DGX OS	Blackwell GPU
Jetson AGX Orin	12× Arm Cortex-A78AE, 2.2 GHz	12	64 GB	Jetson Linux	Ampere GPU
Raspberry Pi 5	Broadcom BCM2712, Cortex-A76 2.4 GHz	4	4 GB	Pi OS 64-bit, 6.18	none
MacBook Air M3	Apple M3, 4P+4E	8	16 GB	macOS 26.6.2	ANE
iPhone 13 Pro	Apple A15 Bionic, 2P+4E	6	6 GB	iOS 26.6.2	ANE
Apple Watch SE 2	Apple S8	2	1 GB	watchOS 26.6	none

governor fixed to performance mode at 2.4 GHz. Each timing run begins below 60 °C. Long-running baseline measurements that reach the thermal limit are annotated separately where applicable.



(a) $\mathcal{N}(b_i)$: rate against flat-table size, Kodak.

(b) Reference-set size N for IPR-L.

Figure 8: **Operating-point sweeps for IPR.** (a) Compression rate versus dense LUT size as the context width n increases on Kodak. The reported setting uses $n = 13$. (b) IPR-L compression rate as the number N of reference images used to construct the LUT increases across four datasets. Each curve is evaluated on a fixed held-out subset, and the reported setting $N = 8$ is highlighted.

A.5 Baseline and Evaluation Details

Classical operating points. Table 1 uses JPEG-XL at effort 7, PNG at level 9, WebP at `z=9/m=6` in lossless mode with the evaluation wrapper's quality setting of zero, JPEG2000 in reversible lossless mode, and FLIF with `-E100`. JPEG-LS is evaluated in lossless mode. Each codec uses one fixed operating point in the main comparison. Table 9 additionally reports PNG levels 1, 3, 5, 7, and 9, WebP settings 0, 2, 4, 6, and 9, and JPEG-XL efforts 1–7 and 9. The operating-point sweep exposes the rate–throughput trade-off beyond the fixed main-table settings.

Learned implementations. The locally evaluated learned baselines use the authors' implementations of L3C, SReC, ArIB-BPS, FNLIC, OmniZip, and FSAR. The model asset sizes and parameter counts of the evaluated configurations are listed in Table 3. For device-side OmniZip measurements, model prediction follows the official implementation, while the remaining coding path is implemented in the common evaluation framework. FSAR is ported to CPU execution for the reported measurements, but not to iOS/Xcode. Unavailable device results remain unreported rather than being extrapolated from other hardware. PILC is included as a reference from the original paper because the source code is unavailable. The PILC reference uses a V100 and 32×32 processing blocks, reports 3.41 bpsp on DIV2K with 180.3/217.2 MB/s encoding/decoding throughput, and is excluded from speedup claims based on locally re-evaluated baselines.

Rate and timing. For an RGB image of width W and height H, bpsp is the accounted number of compressed bits divided by $3WH$ for both 8-bit and 16-bit inputs. Throughput is the uncompressed input byte count divided by wall-clock time, with one byte per 8-bit sample and two bytes per 16-bit sample. Table 1 measures a single image per request without inter-image concurrency. Each measurement uses one warm-up run and 30 timed repetitions of the same image and reports mean throughput. Encoding spans the raw image to the compressed file, and decoding spans the compressed file to the reconstructed raw image. The reversible transforms and entropy coder are included in the timing. The CPU comparisons use the fixed machine-specific worker-thread setting $T^\star$ for the classical and IPR codecs. The separate concurrency sweep shares a 256-thread CPU budget across concurrent images for IPR and classical codecs. Learned baselines use independent GPU processes, except OmniZip, which uses a batch size equal to the number of concurrent images. All reported lossless coding results were verified by byte-exact reconstruction.

Reusable-table calibration. For each dataset, IPR-L constructs the model-derived marginal prior and pooled state counts from the first eight images, then fixes the LUT using Eq. 13 with $\alpha = 32$. The main comparison evaluates the full dataset, including the eight calibration images. The held-out accumulation sweep in Fig. 8(b) uses a separate protocol described in Section A.6.

Table 9: **Rate–throughput comparison across codec operating points on the GPU server.** PNG, WebP, and JPEG-XL are swept over their compression settings and compared with the three IPR variants. Rows are ordered by bpsp to expose throughput differences at similar compression rates. Bold and underlined values denote the best and second-best results in each column, respectively. † marks the setting reported in Table 1.

Codec (setting)	bpsp	Enc. (MB/s)	Dec. (MB/s)
PNG (L1)	4.6247	50.89	170.3
PNG (L5)	4.5266	19.29	168.1
PNG (L3)	4.5258	30.34	174.8
PNG (L7)	4.4678	9.100	168.8
PNG (L9)†	4.4570	7.820	172.5
JPEG-XL (e1)	3.6418	**1,109**	<u>786.7</u>
WebP (z0)	3.4821	82.47	197.5
JPEG-XL (e2)	3.4368	147.3	561.3
WebP (z2)	3.2711	21.48	187.8
WebP (z4)	3.2285	20.98	185.0
WebP (z6)	3.2031	21.00	184.0
WebP (z9)†	3.2031	14.08	206.2
IPR-L *(Ours)*	3.1937	<u>656.5</u>	**826.4**
IPR-L+MDL *(Ours)*	3.1737	53.20	405.1
JPEG-XL (e3)	3.1464	90.83	237.1
IPR-I *(Ours)*	3.1386	0.268	515.5
JPEG-XL (e4)	3.1205	11.66	237.9
JPEG-XL (e5)	3.0586	5.840	179.4
JPEG-XL (e6)	2.9428	3.630	159.9
JPEG-XL (e7)†	<u>2.9020</u>	2.350	164.4
JPEG-XL (e9)	**2.8850**	0.653	190.0

Table 10: **IPR-L prior-mixing ablation.** Rates are in bpsp (lower is better). Bold and underline mark the best and second-best result per dataset; the shaded row uses the selected $\alpha = 32$.

Estimator	Kodak	DIV2K	TESTIMAGES	MIT5K
KT only	3.1965	3.2425	9.9005	11.3668
INR only	3.2066	3.2543	9.9164	**11.3232**
Hybrid ($\alpha = 1$)	3.1951	3.2424	9.8999	11.3605
Hybrid ($\alpha = 8$)	**3.1935**	<u>3.2422</u>	<u>9.8983</u>	11.3478
Hybrid ($\alpha = 32$)	<u>3.1937</u>	**3.2421**	**9.8969**	<u>11.3386</u>

A.6 Additional Experiments

Context width. Fig. 8(a) varies the causal-context width on Kodak, using the hybrid IPR-L estimator in Eq. 13 with $\alpha = 32$. The IPR-L curve uses the first eight reference images and evaluates all 24 images. Dense table sizes are computed from Eq. 16. Increasing the context improves the rate while expanding the lookup space exponentially, as quantified in Eq. 16. The reported setting $n = 13$ provides a favorable rate–memory trade-off before the dense table grows sharply. The plotted IPR-I size denotes the fully materialized domain, rather than the memory occupied by the lazy cache.

LUT calibration. Fig. 8(b) measures the hybrid estimator with $\alpha = 32$ as the number of reference images increases. Each reference image supplies a fitted model and empirical coding events. The fixed evaluation subset contains the last eight Kodak images or the last 16 images of each other dataset. Each curve uses a fixed held-out subset, isolating the calibration-set size from changes in the evaluation images. The largest improvements occur with the first few reference images, followed by diminishing returns on most datasets. Eight reference images provide a favorable calibration–rate trade-off, rather than an exact saturation point. The LUT dimensions remain fixed across the sweep because the state geometry does not depend on the number of accumulated images. The held-out sweep is distinct from the full-dataset main evaluation, which includes the calibration images.

Prior mixing with limited calibration. Fig. 9 varies the calibration size $N \in \{1, 2, 4, 8\}$ while fixing the held-out set to Kodak images 9–24 and DIV2K images 0809–0900. Both the pooled counts and model-derived prior use only the first N reference images, with $\alpha = 32$ and fixed state geometry. The hybrid estimator improves over KT-only at every N: the gain decreases from 0.0440 to 0.0088 bpsp on Kodak and from 0.0054 to 0.0006 bpsp on DIV2K as N increases from one to eight. All aggregate gains have positive 95% confidence intervals from 5,000 paired-image bootstrap

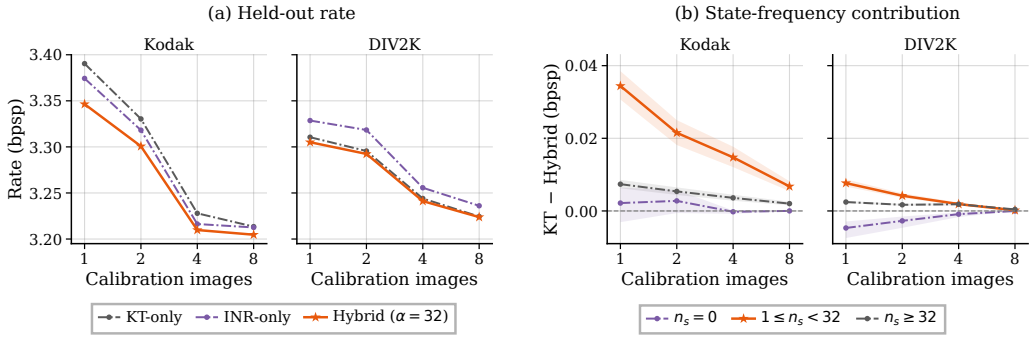


Figure 9: **Prior mixing with limited calibration.** (a) Held-out rates of KT-only, INR-only, and hybrid estimators. (b) State-frequency contributions to the KT-minus-hybrid log-loss, normalized by the number of source subpixels; positive values favor the hybrid estimator. States are grouped by their calibration counts: unseen ($n_s = 0$), sparse observed ($1 \leq n_s \leq 31$), and frequent ($n_s \geq 32$). Shaded bands show paired-image bootstrap 95% confidence intervals.

resamples, conditional on the fixed calibration sets. For Fig. 9(b), n_s counts coding events at state s in the N calibration images, not in the held-out set. At each N, we partition states into unseen ($n_s = 0$), sparse observed ($1 \leq n_s \leq 31$), and frequent ($n_s \geq 32$) groups; group membership is recomputed as calibration grows. Each curve aggregates all held-out coding events whose states belong to that group, rather than plotting individual states or averaging them with equal weight. For a group G, its contribution is $\sum_{i:s_i \in G} \log_2[p_{\text{Hybrid}}(b_i \mid s_i)/p_{\text{KT}}(b_i \mid s_i)]$ divided by the total number of source subpixels in the held-out set. The three contributions therefore sum to the total log-loss improvement; they are not conditional bits per event within each group and exclude coder and header overhead. States observed 1–31 times benefit on both datasets at every N, accounting for 72–81% of the total log-loss gain on Kodak; frequent states contribute more than this sparse group on DIV2K at $N = 8$. For unseen states, the gain intervals include zero on Kodak and are negative on DIV2K at every N. The positive sparse-group contributions are consistent with the INR prior stabilizing probability estimates based on few observations. These results support regularization of sparsely observed states rather than a consistent gain on unseen states.

Rate–throughput trade-off. Table 9 sweeps PNG, WebP, and JPEG-XL settings on the GPU server. IPR-L achieves 3.1937 bpsp at 656.5/826.4 MB/s encoding/decoding throughput. Compared with JPEG-XL effort 2, IPR-L has a lower rate and higher throughput in both directions. JPEG-XL effort 3 reaches a slightly lower rate but substantially lower throughput. WebP `z 6` reaches a nearly identical rate of 3.2031 bpsp with 21.00/184.0 MB/s throughput. IPR-I combines a lower rate of 3.1386 bpsp with 515.5 MB/s decoding, while per-image fitting limits encoding to 0.268 MB/s.

Prior-mixing ablation Table 10 compares the count-based KT estimator, the INR-derived prior, and the hybrid estimator in Eq. 13. The hybrid estimator improves over KT-only estimation across all four datasets. Increasing α strengthens the contribution of the model-derived prior, with $\alpha = 32$ selected for the reported IPR-L results. The INR-only variant performs worse than the selected hybrid estimator on Kodak, DIV2K, and TESTIMAGES, while achieving a lower rate on MIT5K.