FRAPPE: Full Input, Residual Output Autoencoding with Projection Pursuit Encoder

Dan Jacobellis
Dept. of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712, USA
danjacobellis@utexas.edu
ORCID: 0000-0001-8541-1906

Neeraja J. Yadwadkar
Dept. of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712, USA
neeraja@austin.utexas.edu
ORCID: 0009-0007-7556-3069

***Abstract*—Media compression standards have reached a plateau in terms of the rate-distortion-complexity trade-off, limiting the ability to offload expensive AI perception to the cloud in applications like robotics, wearables, and remote sensing. DNN-based codecs improve compression efficiency, but at a cost: they cannot easily adapt to large changes in available bitrate, and real-time encoding requires expensive, power-hungry GPUs that prohibit use on low-cost or resource-constrained platforms. To address these limitations, we propose a novel autoencoding framework (FRAPPE) that uses the Full input to predict the Residual output via a Projection-Pursuit Encoder. FRAPPE's encoding objective naturally sorts latent channels by importance, allowing zero-overhead variable-rate coding. Unlike RNN-based learned codecs, whose encoder consumes the previous reconstruction's residual, or RVQ-style codecs, whose codebooks must be applied sequentially, FRAPPE's analysis path is an embarrassingly parallel DAG of independent input projections. Using FRAPPE, we build a variable-rate RGB image codec (FRAPPE-Image), and evaluate its rate-distortion-complexity trade-off against standard image codecs. At high compression ratios ($\sim$0.1 bpp) FRAPPE-Image provides higher perceptual quality than AVIF with 47$\times$ faster encoding, making it capable of real-time 1080p, 30fps CPU-only encoding. Our code and pre-trained models are available: https://github.com/UT-SysML/FRAPPE.**

***Index Terms*—data compression, deep learning**

I. Introduction

Current media compression standards like VVC and AV1 have reached a plateau in terms of the rate-distortion-complexity trade-off. Since the standardization of digital media codecs like JPEG and MP3 more than three decades ago, codec design innovations have led to dramatic improvements in signal quality for the same bitrate. However, these increasingly complex designs are burdened by equally dramatic increases in encoding cost and power consumption [1]. For this reason, simpler codecs like JPEG and MP3 remain ubiquitous for power-constrained sensors [2]. For many applications, particularly those involving robotics or wearables, this has severely limited the ability to offload computation to the cloud. Existing codecs all fail in at least one of three ways. (1) They require prohibitively high encoding resources (FLOPS, memory bandwidth, etc.). (2) They provide inadequate compression ratios to transmit data over the cellular, satellite, or BLE communication channels available in the field. (3) They introduce too much distortion or latency to benefit from cloud-based processing. Recent advances in deep neural network (DNN)–based autoencoders [3], [4] have shown potential to break free of this plateau, but make significant compromises in at least one of three dimensions: (1) on-the-fly rate adaptation comparable to standards like JPEG or AVIF; (2) encoding cost competitive with standardized codecs at matched compression efficiency; (3) real-time encoding on commodity hardware without GPU or NPU accelerators for standard-resolution audio, image, or video streams.

To address these issues and improve the utility of cloud-assisted robotics and wearable applications, we propose a new type of residual autoencoder (FRAPPE). FRAPPE uses the **F**ull input to predict the **R**esidual output via a **P**rojection-**P**ursuit **E**ncoder. By using a projection pursuit encoding scheme, FRAPPE sorts the latent channels by importance, allowing zero-overhead variable-rate and progressive coding using a single set of encoder weights. Unlike RNN-based learned codecs [5], [6], whose encoder consumes the previous reconstruction's residual at each iteration, making them prohibitively expensive, and RVQ-style codecs [7]–[9], whose codebooks must be applied sequentially, FRAPPE formulates the residual autoencoding objective using the *full input* to predict the *residual output*. This decouples the per-channel projections so the analysis path becomes a DAG: all latent channels are encoded in parallel and the encoder collapses to S strided convolutions at inference, without any recurrence or quantizer chain. Our contributions are threefold.

- We propose FRAPPE, an autoencoding framework designed to provide (1) variable rate and progressive compression, (2) competitive rate distortion performance at high compression ratios, and (3) low encoding costs to enable use with resource constrained sensors
- Using this framework, we instantiate and train a practical image compression system.
- We evaluate FRAPPE-Image against other conventional and learned image codecs and demonstrate extreme gains in terms of the rate-distortion-complexity trade-off.

Background and related work. FRAPPE builds upon previous works related to asymmetric neural codec design, residual

autoencoding, and projection pursuit algorithms.

Asymmetric neural codec design. The asymmetric design philosophy of WaLLoC [3] and LiVeAction [4]—a heavy nonlinear synthesis transform paired with a deliberately lightweight analysis transform—is well suited to resource-constrained encoding. FRAPPE inherits this stance, along with the log-variance rate proxy used by LiVeAction. MCU-Coder [2] achieves encoding efficiency gains in an asymmetric architecture using post-training quantization.

Residual autoencoding. The closest neural-codec analogues are the Toderici et al. recurrent compressors [5], [6], which encode an image as a chain of additive reconstructions $\hat{x}_t = \hat{x}_{t-1} + D_t(E_t(r_{t-1}))$ in which each stage's encoder consumes the previous reconstruction's residual, requiring the decoder to be evaluated inside the encoding loop. Neural audio codecs built on residual vector quantization [7]–[9] avoid this by pushing the residual recursion into the quantizer chain instead, but the chain itself remains sequential at encode time. More broadly, fitting a sum of terms one at a time on the residual of the preceding fit is the forward stagewise additive modeling framework [10], of which projection-pursuit regression is the supervised special case. Classical signal-processing precursors include matching pursuit [11] and orthogonal matching pursuit [12]—greedy dictionary expansions whose atom-selection rule is itself recognized as a special case of projection pursuit [12]—alongside the cascade-correlation constructive network [13] and greedy layer-wise autoencoder pretraining [14], which add components one at a time but operate on a latent rather than an output residual. Closest in spirit to FRAPPE's deflation pattern are alternating least squares for nonlinear PCA [15], [16], deflation-based canonical correlation analysis [17], [18], and one-unit deflation-mode FastICA [19], [20], the last of which explicitly identifies each extracted direction with a projection-pursuit index.

Projection pursuit. Projection pursuit [21], [22] is a family of methods for finding informative linear projections $\hat{k}^\top X$ of multivariate data by varying the projection direction $\hat{k}$ so as to maximize a continuous index of "usefulness." In the original algorithm formulation [21], unconstrained hill-climbing is applied to a smoothed index measuring the product of global spread and local density in the projected dimension, producing multiple distinct projections by restarting from different seeds and constraining subsequent searches to subspaces orthogonal to already-found directions. Projection pursuit regression (PPR) [23] extends the method to supervised learning by fitting the additive model

$$f(X) = \sum_{m=1}^{M} g_m(\omega_m^\top X), \tag{1}$$

where each ω_m is a learned unit projection direction and each g_m is a nonlinear function. PPR is fit forward-stagewise: at stage m, a new pair (ω_m, g_m) is added to minimize the residual error left by the previous $m-1$ components, and prior directions are typically frozen [22]. The number of components M is determined by the stagewise procedure itself: fitting terminates when the next term no longer appreciably improves the fit.

II. Proposed Method

To enable real-time, cloud-assisted machine perception on the resource-constrained sensors used in robotics and wearables, FRAPPE is designed around three goals: (1) zero-overhead variable-rate and progressive coding with a *single* set of encoder weights; (2) rate–distortion performance competitive with standardized codecs (JPEG, AVIF); (3) high-throughput encoding on low-power sensors without GPUs or accelerators.

Codec workflow. Let $x \in \mathbb{R}^{C\times T_1\times\cdots\times T_D}$ denote a signal with C channels and $D \in \{1,2,3\}$ spatio-temporal dimensions, normalized to $[-1,1]$. FRAPPE composes an analysis transform $\mathcal{G}_A$, an entropy-coded quantizer $\mathcal{Q}$, and a synthesis transform $\mathcal{G}_S$:

$$\hat{x} = \mathcal{G}_S \circ \mathrm{Adapt}_{p_d} \circ \mathcal{Q} \circ \Phi \circ \mathcal{G}_A(x). \tag{2}$$

The analysis transform $\mathcal{G}_A$ splits into S scale groups, where group s carries n_s latent channels at patch size p_s; each channel is a single learned linear projection of a non-overlapping patch of x. The companding nonlinearity Φ confines every channel to a signed 8-bit range; the quantizer $\mathcal{Q}$ rounds to integers and per-scale latents are entropy-coded independently. Before reconstruction, Adapt_{p_d} rebins each scale's grid to a common decoder resolution p_d and the resulting tensors are concatenated for $\mathcal{G}_S$. The trained instance evaluated in Section III (henceforth FRAPPE-Image) operates on RGB images (C=3, D=2) and uses S=5 scale groups with $(n_s, p_s) = (3,32), (6,16), (3,8), (6,4), (3,2)$ for N=21 latent channels total, and p_d=8.

(a) Residual autoencoding with a progressively relaxing entropy bottleneck. FRAPPE introduces channels one at a time in coarse-to-fine order. Let $\mathcal{F}_{m-1}$ denote the merged codec over the first $m-1$ channels (with $\mathcal{F}_0 \equiv 0$). The m-th channel's encoder–decoder pair is fit to the *output-space* residual $r_m = x - \mathcal{F}_{m-1}(x)$ by minimizing

$$\begin{aligned}\mathcal{L}_m &= \log_{10}\left\|r_m - \hat{r}_m\right\|_2^2 \\ &\quad + \lambda_m \left(\mathbb{E}\, r_m^2\right)^{\rho} \log_2 \mathrm{Std}\big(\Phi(\omega_m^\top \mathrm{Patch}_{p_m}(x))\big),\end{aligned} \tag{3}$$

where ω_m is the new channel's projection direction, $\hat{r}_m$ is the single-channel autoencoder's prediction, and the second term is LiVeAction's log-variance rate proxy [4] re-weighted by the (detached) residual power $\mathbb{E}\, r_m^2$ raised to ρ=0.3; without this re-weighting the rate term grows to dominate the distortion term as residual energy decays, collapsing later channels to near-zero output.

The patch size and the Lagrangian λ_m relax monotonically across the sequence. Channel 0 has the most aggressive bottleneck: at patch size p_0 a single 8-bit latent samples a Cp_0^D-dimensional input patch, paired with the largest λ_m (for FRAPPE-Image, 3×32^2=3072, a 3072:1 per-channel dimensionality reduction). By the final channel the bottleneck has relaxed to Cp^D at the finest patch size and a smaller λ_m (for

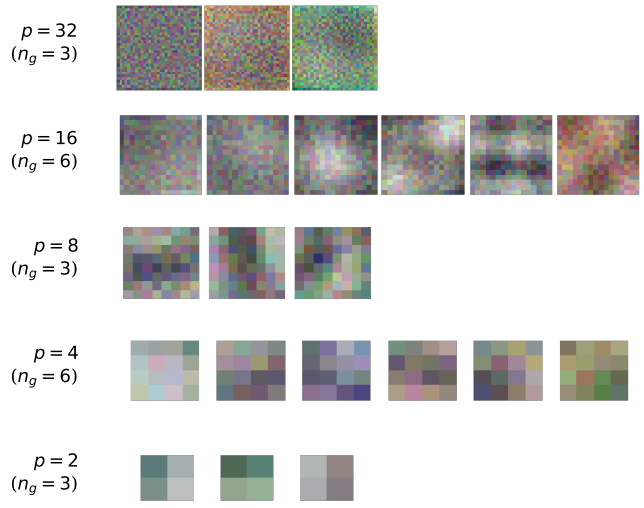

Fig. 1. Consolidated encoder weights of FRAPPE-Image, one row per scale group. Each tile is a learned filter $\omega_m \in \mathbb{R}^{3\times p_s\times p_s}$ rendered as RGB, normalized to $\pm 4\sigma$ within its scale row. The five rows show $(n_s, p_s) = (3,32),(6,16),(3,8),(6,4),(3,2)$ for N=21 channels. When trained on sRGB inputs, FRAPPE-Image learns, without supervision, a representation similar to chroma subsampling in a luma, chrominance-orange, chrominance-green (YCoCg) color space.

FRAPPE-Image, channel 20 at p_{20}=2 gives 12:1). Each new channel only needs to capture variance not already explained by its predecessors, so the schedule yields latents that are naturally sorted by importance with no explicit decorrelation loss. Fig. 1 shows the resulting filters: coarse channels carry low-frequency luma/chroma DC, finer scales sort into oriented edges and color textures. This intrinsic ordering directly delivers goal (1): retaining only the first n channels and selecting the matching merged-decoder snapshot recovers a rate point on the operating curve, with no auxiliary scale-selection module, fine-tuning, or encoder rerun. Fig. 2 traces this sweep on a single Kodak image, and Fig. 3 (Section III) reports the full curves over the Kodak set.

(b) Asymmetric design via full-input projection-pursuit encoding. DNN-based autoencoders earn their rate–distortion advantage by leveraging large datasets and substantial decode-time compute. FRAPPE targets an asymmetric deployment topology: capture-side encoding on resource-constrained sensors, cloud-side decoding on workstation hardware. This inverts the encode-once/decode-many model of broadcast media (VVC, AV1, HEVC), where decoder cost is the binding constraint; here it is paid once per upload at the cloud, which can transcode to formats suitable for downstream applications. We therefore adopt the asymmetric philosophy of WaLLoC [3] and LiVeAction [4]—a powerful nonlinear synthesis transform paired with a deliberately lightweight analysis transform—and combine it with the residual scheme of (a) by a specific design choice: each channel's encoder operates on the *full input* x rather than on the latent-space residual of previous channels. The training target stays the output-space residual r_m, but the encoder input does not.

This realizes the projection-pursuit regression model of Section I (cf. Eq. (1)). Channel m takes

$$z_m \;=\; \Phi\big(\omega_m^\top \operatorname{Patch}_{p_m}(x)\big), \tag{4}$$

with $\omega_m \in \mathbb{R}^{C\,p_m^D}$ a learned projection direction and the per-channel ridge function g_m realized jointly by the merged synthesis transform across all channels. Because all n_s channels in scale group s share the same patch size and the same input, their projections consolidate into a single strided convolution at inference,

$$z^{(s)} \;=\; \Phi\big(W^{(s)} *_{p_s} x + b^{(s)}\big), \quad W^{(s)} \in \mathbb{R}^{n_s\times C\times p_s^{(D)}}, \tag{5}$$

where $p_s^{(D)}$ denotes the D-fold product $p_s\times\cdots\times p_s$ and $*_{p_s}$ is D-dimensional strided convolution with stride p_s. The consolidation is exact—the channels were trained one at a time but never share a kernel or interact across channels in the analysis path—so the FRAPPE-Image encoder is just S=5 `Conv2d` layers followed by per-channel companding and quantization.

The synthesis transform absorbs nearly all the model's parameters and FLOPs. Its architecture is fixed across channel counts (only the first pointwise projection's input widens with n), but its weights are snapshotted: one retrained $\mathcal{G}_S$ per supported channel count. Because all encoders are frozen during this retraining, encoder weights are bit-identical across snapshots, and a single set of encoder weights serves every n. The body is a kernel-3 projection to a fixed width, a stack of ConvNeXt-style [24] residual blocks (depthwise kernel-3, LayerNorm, pointwise expand by $4\times$, GELU, pointwise contract, with LayerScale), a pointwise projection to Cp_d^D channels, a stride-p_d transposed D-dimensional convolution, and Hardtanh; FRAPPE-Image instantiates the stack at width 768 with twelve blocks. Each scale group's quantized latents are first remapped to the decoder resolution p_d,

$$\operatorname{Adapt}_{p_d}\big(z^{(s)}\big) \;=\; \begin{cases} \operatorname{S2D}_{p_d/p_s}\big(z^{(s)}\big), & p_s < p_d, \\ z^{(s)}, & p_s = p_d, \\ \operatorname{NN}_{p_s/p_d}\big(z^{(s)}\big), & p_s > p_d, \end{cases} \tag{6}$$

where $\operatorname{S2D}_f$ folds f^D-sample blocks into the channel dimension (one encoder channel becomes f^D decoder channels) and NN_f is nearest-neighbor upsampling. The adapted tensors are concatenated and fed to $\mathcal{G}_S$. With $C_d = \sum_{p_s\le p_d} n_s (p_d/p_s)^D + \sum_{p_s>p_d} n_s$ adapted decoder-input channels (for FRAPPE-Image, $C_d = 3+6+3+24+48 = 84$), (2) expands to

$$\hat{x} \;=\; \mathcal{G}_S\Big(\bigoplus_{s=1}^{S} \operatorname{Adapt}_{p_d}\big(\mathcal{Q}\,\Phi(W^{(s)} *_{p_s} x + b^{(s)})\big)\Big), \tag{7}$$

with $\bigoplus$ denoting channel-wise concatenation.

(c) Cheap, parallelizable analysis transform. Because the analysis path consists only of a strided convolution and a pointwise nonlinearity, its per-sample cost is closed-form. The strided convolution from C input channels to N latent channels touches each input sample exactly once and contributes

Fig. 2. Progressive reconstructions of `kodim22` as the transmitted channel count n is varied. All n panels share the same encoder weights; only the truncated channel count and matching merged-decoder snapshot differ. The bottom-right panel is the uncoded reference. Bits-per-pixel measurements are JPEG-LS-coded.

CN multiply–accumulates per sample *regardless of patch size*—a patch of size p_s^D requires Cp_s^D MACs but covers p_s^D samples, so the per-sample cost is C MACs per channel. The softsign compander $\Phi_c(u) = ru/(\sigma_c + |u|)$ with r=127 guarantees $|\Phi_c(u)| < r$ and so fits the companded activations into a signed 8-bit range. The denominator scale σ_c is learned per latent channel, and a learned per-channel multiplier is applied to the output (one scalar each per channel); together they cost 4 operations per latent element (absolute value, addition, division, post-softsign multiply; the fixed scalar r fuses into the divide). Per sample, scale group s contributes only $4n_s/p_s^D$ companding ops. The full analysis path therefore costs $CN + \sum_s 4n_s/p_s^D$ ops per sample, dominated by the linear projection and independent of decoder depth or number of scale groups; for FRAPPE-Image this evaluates to ≈68 ops/pixel, with even the finest scale (n_s=3, p_s=2) adding just 3 ops/pixel.

Equally important, the per-scale strided convolutions in (5) share the input x but are otherwise independent, so the analysis path forms an unconstrained DAG whose nodes can be pipelined or evaluated in parallel—there is no recurrent encoder dependency [5], [6] and no sequential residual-quantizer chain [7]–[9] to serialize the encode pass. After companding and rounding, scale group s produces $\mathcal{Q}(z^{(s)}) \in \mathbb{Z}^{n_s \times T_1/p_s \times \cdots \times T_D/p_s}$, which is serialized per scale and concatenated into the full bitstream; per-scale coding is the natural choice given that scales have different spatial resolutions. Pre-quantization activations approximately follow a generalized Gaussian distribution close to a Laplacian, as is typical of subband coefficients of natural signals [25], so any 8-bit lossless codec whose prediction residuals are modeled with a Laplacian-like distribution is nearly entropy-optimal. The implementation isolates entropy coding behind a four-function contract so any modality-appropriate lossless codec can be substituted (e.g. FLAC for 1D signals); FRAPPE-Image reshapes each scale to a single 2D grayscale plane $(n_s \cdot T_1/p_s, T_2/p_s)$ and applies length-prefixed JPEG-LS [26], whose Golomb–Rice prediction residuals are two-sided geometric—the discrete analog of a Laplacian.

Training and implementation details. We train FRAPPE-Image on the LSDIR dataset [27] with batch size 1 using the Adan optimizer [28]; Kodak is held out for validation. Each channel passes through two stages. The single-channel residual stage fits (ω_m, g_m) at peak learning rate 1.5×10^{-5} on a steep cosine ramp; the small peak reflects that the encoder is being adapted to a residual that $\mathcal{G}_S$ already partially explains. After fitting, the new encoder weights are merged into their scale group, all m encoders are frozen, $\mathcal{Q}$ is switched from training-time additive noise to hard rounding, and only $\mathcal{G}_S$ is retrained on the union of latents (with λ=0) at peak learning rate 5×10^{-4} on a milder ramp. Within either stage the encoder parameter group runs at one-tenth the decoder learning rate, keeping the lightweight projections stable while the heavier synthesis transform absorbs most of the optimization signal. Per-channel epoch counts ramp coarse-to-fine (single-channel 2→7, merged-decoder 4→7), reflecting that later channels carry smaller residual energy and finer detail. The full per-channel λ_m schedule and training scripts are available in the accompanying code repository[1].

III. Experimental Data and Results

We evaluate the rate-distortion-complexity performance of FRAPPE-Image on the Kodak dataset. We compare against conventional transform codecs (JPEG, AVIF) as well as symmetric and asymmetric neural codecs (mbt2018 [29] and WaLLoC [3], respectively) on a shared CPU testbed (AMD EPYC 9354). Rate is measured using bits per pixel (bpp), where 24 bpp corresponds to 8-bit RGB inputs. Distortion is measured using conventional and perceptual metrics (PSNR, SSIM [30], and DISTS [31]) at the original image resolution of 768×512 or 512×768. Following [3], DISTS is reported in decibels as $\text{DISTS}_{\text{dB}} = -10\log_{10}(\text{DISTS})$ so that higher values indicate better perceptual quality. Consistent with FRAPPE's asymmetric deployment topology (Section II), we report encoder-

[1] https://github.com/UT-SysML/FRAPPE

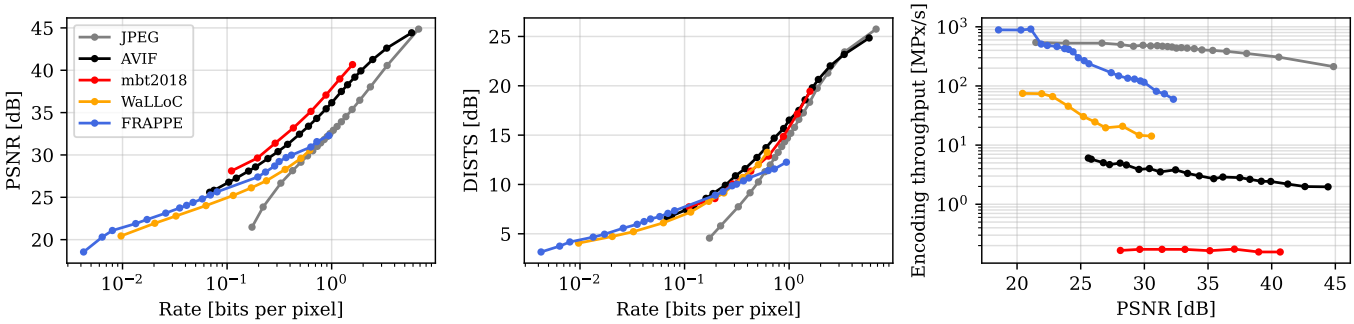

Fig. 3. Rate-distortion and encoding-throughput comparison on Kodak for JPEG and AVIF (both via Pillow), mbt2018 (via CompressAI), WaLLoC, and FRAPPE-Image. Left and middle: PSNR and DISTS vs rate (bits per pixel, log scale). Right: encoding throughput (MPx/s, log scale) vs PSNR.

side throughput, measured as the median over five timed runs (one warmup) on CPU and timed end-to-end through the analysis transform, companding/quantization, and JPEG-LS entropy coding; no GPUs or hardware accelerators are used at inference for any codec. AVIF results use Pillow over libavif at default speed and effort with no tile or thread tuning—the configuration most production deployments rely on. Fig. 3 compares the rate–distortion–complexity trade-off of FRAPPE-Image against JPEG, AVIF, mbt2018, and WaLLoC; additional measurements against JPEG XL, LiVe-Action [4], and MCUCoder [2] are reported in the appendix, with FRAPPE-Image holding a $+3.1$ to $+4.3$ dB BD-PSNR lead over MCUCoder at matched rate.

Exceptional performance at high compression ratios. At bitrates near 0.1 bpp (compression ratio of 240:1) FRAPPE-Image provides better perceptual quality (DISTS) than AVIF and 47 times faster encoding. The advantage extends across the low-rate band: FRAPPE attains the best mean BD-DISTS in every regime below 0.215 bpp against every baseline in Table I.

Real-time CPU-only encoding. FRAPPE-Image is capable of real-time (1080p, 30fps) CPU encoding even at high quality levels (n=20, roughly 31.5 dB PSNR). In comparison, DCVC-RT [32], the first neural video codec capable of real-time encoding, requires a high-power GPU to reach similar throughput and does not support CPU inference.

Extreme compression ratios. FRAPPE-Image can provide extreme compression ratios in excess of 5000:1, while the lowest AVIF and JPEG settings only reach 352:1 and 139:1, respectively. Among the learned baselines only WaLLoC reaches the sub-25 dB PSNR regime FRAPPE targets at these ratios; mbt2018's quality grid bottoms at 28 dB and is therefore absent from the lowest two PSNR regimes of Table II.

Fixed quality target. For a fixed quality target of 21 dB PSNR, FRAPPE-Image encodes 1.7 times faster than JPEG (915 MP/sec vs 544 MP/sec) while providing 22 times higher compression ratio (0.0080 bpp vs 0.173 bpp). AVIF's throughput on the same CPU testbed ranges from 1.97 to 6.04 MP/sec.

PSNR/SSIM lead of mbt2018 comes at a steep throughput cost. mbt2018 retains a BD-PSNR advantage of $+2.2$ to $+4.2$ dB over FRAPPE-Image across the $[0.1, 1)$ bpp band (Table I), but at 0.16–0.17 MPx/s on the same CPU testbed—up to $\sim$1000$\times$ slower than FRAPPE's encode throughput (74–168 MPx/s) at matched rates. The PSNR-optimal regime mbt2018 dominates is therefore unreachable in the asymmetric, on-sensor encoding setting that motivates FRAPPE.

IV. Conclusion

We presented FRAPPE, a powerful representation-learning technique suitable for zero-overhead variable-rate lossy compression on resource-constrained sensors. Using this framework, we built a practical image compression system, FRAPPE-Image, which performs favorably against existing codecs in terms of the trade-off between rate, distortion, and encoding complexity.

Limitations and future work. The framework applies to any 1D, 2D, or 3D signal with an arbitrary channel count, but our experiments cover only RGB images; instantiations for audio, hyperspectral images, video, and 3D volumes are an obvious extension. FRAPPE-Image is intentionally biased toward low-rate, perceptual-quality operating points and the encoder-side resource budget; at moderate-to-high bitrates conventional symmetric codecs and learned baselines with heavier analysis transforms retain a rate–distortion advantage on PSNR/SSIM, and our experiments do not include hyperprior, autoregressive, or recent variable-rate learned codecs (e.g. conditional, prompt-tuned, or quantizer-tuning approaches)—a head-to-head against these on the same CPU testbed is left to future work. Variable-rate operation here is realized by storing one merged-decoder snapshot per supported channel count n (21 snapshots in FRAPPE-Image), which is a substantial storage and deployment burden; training a single decoder with random channel dropout [2] to handle arbitrary channel subsets is a natural next step. Broader datasets (Tecnick, CLIC), higher resolutions, libaom-av1 with tuned speed presets, and ablations over the compander, ρ, and the λ_m schedule are all left to a longer companion paper. The entropy stage (JPEG-LS over companded 8-bit latents) is deliberately simple and CPU-friendly; substituting a learned or per-image entropy model is straightforward within our four-function entropy contract and could close part of the rate gap at moderate bitrates without changing the encoder.

References

[1] F. Bossen, K. Sühring, A. Wieckowski, and S. Liu, "VVC complexity and software implementation analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3765–3778, 2021.

[2] A. Hojjat, J. Haberer, and O. Landsiedel, "Mcucoder: Adaptive bitrate learned video compression for IoT devices," in *DAGM German Conference on Pattern Recognition*. Springer, 2025, pp. 123–138.

[3] D. Jacobellis and N. J. Yadwadkar, "Learned compression for compressed learning," in *2025 Data Compression Conference (DCC)*. IEEE, 2025.

[4] D. Jacobellis and N. J. Yadwadkar, "LiVeAction: Lightweight, versatile, and asymmetric codec design for real-time operation," in *IEEE Data Compression Conference (DCC)*, 2026, in press. [Online]. Available: https://ut-sysml.github.io/liveaction

[5] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar, "Variable rate image compression with recurrent neural networks," *arXiv preprint arXiv:1511.06085*, 2015.

[6] G. Toderici, D. Vincent, N. Johnston, S. Jin Hwang, D. Minnen, J. Shor, and M. Covell, "Full resolution image compression with recurrent neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 5306–5314.

[7] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, "SoundStream: An end-to-end neural audio codec," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 495–507, 2022.

[8] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "High fidelity neural audio compression," *Transactions on Machine Learning Research*, 2023.

[9] R. Kumar, P. Seetharaman, A. Luebs, I. Kumar, and K. Kumar, "High-fidelity audio compression with improved RVQGAN," in *NeurIPS*, 2023.

[10] T. Hastie, R. Tibshirani, and J. Friedman, "Forward stagewise additive modeling," in *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed., ser. Springer Series in Statistics. New York: Springer, 2009, ch. 10.2, pp. 389–392.

[11] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on signal processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[12] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proceedings of 27th Asilomar conference on signals, systems and computers*. IEEE, 1993, pp. 40–44.

[13] S. Fahlman, "The recurrent cascade-correlation architecture," *Advances in neural information processing systems*, vol. 3, 1990.

[14] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, 2006.

[15] F. W. Young, Y. Takane, and J. de Leeuw, "The principal components of mixed measurement level multivariate data: An alternating least squares method with optimal scaling features," *Psychometrika*, vol. 43, no. 2, pp. 279–281, 1978.

[16] G. Michailidis and J. De Leeuw, "The gifi system of descriptive multivariate analysis," *Statistical Science*, pp. 307–336, 1998.

[17] T. R. Knapp, "Canonical correlation analysis: A general parametric significance-testing system." *Psychological Bulletin*, vol. 85, no. 2, p. 410, 1978.

[18] W. K. Härdle and L. Simar, "Chapter 16: Canonical correlation analysis," in *Applied multivariate statistical analysis*. Springer, 2015.

[19] A. Hyvärinen and E. Oja, "A fast fixed-point algorithm for independent component analysis," *Neural computation*, vol. 9, no. 7, pp. 1483–1492, 1997.

[20] A. Hyvarinen, "Fast and robust fixed-point algorithms for independent component analysis," *IEEE transactions on Neural Networks*, vol. 10, no. 3, pp. 626–634, 1999.

[21] J. Friedman and J. Tukey, "A projection pursuit algorithm for exploratory data analysis," *IEEE Transactions on Computers*, vol. 23, no. 9, pp. 881–890, 1974.

[22] T. Hastie, R. Tibshirani, and J. Friedman, "Projection pursuit regression," in *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed., ser. Springer Series in Statistics. New York: Springer, 2009, ch. 11.2, pp. 389–392.

[23] J. H. Friedman and W. Stuetzle, "Projection pursuit regression," *Journal of the American statistical Association*, vol. 76, no. 376, pp. 817–823, 1981.

[24] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," in *CVPR*, 2022.

[25] K. Sharifi and A. Leon-Garcia, "Estimation of shape parameter for generalized Gaussian distributions in subband decompositions of video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, no. 1, pp. 52–56, 1995.

[26] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS," *IEEE Transactions on Image Processing*, vol. 9, no. 8, pp. 1309–1324, 2000.

[27] Y. Li, K. Zhang, J. Liang, J. Cao, C. Liu, R. Gong, Y. Zhang, H. Tang, Y. Liu, D. Demandolx *et al.*, "Lsdir: A large scale dataset for image restoration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 1775–1787.

[28] X. Xie, P. Zhou, H. Li, Z. Lin, and S. Yan, "Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 12, pp. 9508–9520, 2024.

[29] D. Minnen, J. Ballé, and G. D. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," in *NeurIPS*, 2018.

[30] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[31] K. Ding, K. Ma, S. Wang, and E. P. Simoncelli, "Image quality assessment: Unifying structure and texture similarity," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 5, pp. 2567–2581, 2020.

[32] Z. Jia, B. Li, J. Li, W. Xie, L. Qi, H. Li, and Y. Lu, "Towards practical real-time neural video compression," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 12 543–12 552.

[33] G. Bjøntegaard, "Improvements of the BD-PSNR model," ITU-T SG16/Q6, Document VCEG-AI11, Berlin, Germany, Tech. Rep., 2008.

APPENDIX A
REGIME-LOCALIZED BJONTEGAARD-DELTA ANALYSIS

Tables I and II summarize the operating points of Fig. 3 from two complementary viewpoints, extended with three additional CPU-only baselines: JPEG XL (libjxl, effort=7), LiVeAction [4] (the published `lsdir_f16c48` checkpoint), and MCUCoder [2] (the published MS-SSIM checkpoint via fp32 PyTorch; reported throughput is therefore an upper bound on the deployed INT8/CMSIS-NN encoder). Both anchor the comparison on FRAPPE-Image and prune to a single representative point per (codec, regime) pair, taken at the regime's median value of the binning axis. "Setting" is the codec's sweep parameter: JPEG and AVIF Pillow quality, mbt2018 and WaLLoC integer quality, and FRAPPE-Image transmitted channel count n. Distortion values are means across the 24 Kodak images; throughput is the median over five timed runs (one warmup) on the AMD EPYC 9354 CPU testbed of Section III. Bjontegaard-Delta values [33] are computed via PCHIP interpolation on a window comprising the regime plus one immediately adjacent point on either side; entries marked "–" are regimes where FRAPPE and the test codec curves do not overlap sufficiently along the integration axis. FRAPPE rows are zero by construction.

Table I bins by rate (1/3-decade bpp regimes above 0.0464 bpp, factor $10^{1/3} \approx 2.15$, with all lower-rate operating points collapsed into a single < 0.0464 regime) and reports BD-Metric (BD-PSNR / BD-SSIM / BD-DISTS), the average distortion difference at matched rate. Positive entries indicate the test codec achieves higher quality than FRAPPE in that rate regime. Table II instead bins by quality (PSNR in 2.5 dB regimes from 22.5 to 32.5 dB) and reports BD-Rate, the average percentage rate difference at matched distortion. Negative entries indicate the test codec needs less rate than FRAPPE to match the same quality. The two views are duals: each guarantees overlap along its respective integration axis by construction, eliminating the disjoint-curve regime in which BD-statistics would otherwise be undefined.

APPENDIX B
EVALUATION METHODOLOGY DETAILS

This appendix documents harness-level choices in the open-source evaluation pipeline that materially affect the numbers in Section III and Appendix A.

Throughput vs rate-distortion input shape. Rate-distortion metrics use the native Kodak resolution (768×512 or 512×768); encoder throughput is measured on 512×512 center crops, sidestepping per-codec divisibility constraints (mbt2018 requires multiples of 64) and keeping the throughput denominator constant across codecs. Input pre-staging is excluded from the timer.

Single-threaded CPU. All CPU encodes (Pillow JPEG and AVIF, mbt2018, WaLLoC, FRAPPE) run with `torch.set_num_threads(1)`; Pillow's libavif backend is at default speed and effort with no tile or thread tuning. Reported throughputs are per-thread.

mbt2018 bitstream. The vendored mbt2018 baseline reports bpp from forward-pass likelihoods ($-\sum \log_2 p / n_{\text{pixels}}$) rather than from a real bitstream—the autoregressive context-model `compress()` loop is not invoked. Likelihood-based bpp is a tight lower bound on what an entropy coder over the same likelihoods would achieve, but real CPU encode time would be substantially higher than the forward-pass throughput plotted here, since the autoregressive serialization dominates on CPU. The reported mbt2018 throughput should therefore be read as an upper bound on a deployable encoder.

WaLLoC variable-rate. WaLLoC's quality parameter is a bicubic resize-down applied inside the encoder before the wavelet and learned analysis transforms. The resize cost is included in WaLLoC's reported throughput; the bpp denominator is the original (pre-resize) pixel count, matching the user-facing rate.

FRAPPE encode timing. FRAPPE's throughput is end-to-end through encoder forward pass plus int8 quantization, device-to-host transfer of the quantized latents, and CPU-side latent arrangement plus JPEG-LS entropy coding. Each measurement is one untimed warmup epoch over the 24-image dataset followed by five timed epochs; throughput is megapixels per image divided by the median per-image total time.

TABLE I
RATE-BINNED BD-METRIC ON KODAK

Regime [bpp]	Codec	Setting	bpp	PSNR (dB)	SSIM	DISTS (dB)	Thr. (MPx/s)	BD-PSNR (dB)	BD-SSIM	BD-DISTS (dB)
< 0.0464	WaLLoC	$q = 2$	0.02035	21.93	0.5866	4.73	74.02	-0.80	-0.0475	-0.51
	LiVeAction	$q = 4$	0.03406	22.98	0.6376	5.13	1.90	-0.74	-0.0540	-0.83
	FRAPPE	$n = 4$	0.01337	21.90	0.5870	4.66	**510.48**	**0.00**	**0.0000**	**0.00**
$[0.0464,\ 0.1)$	AVIF	$q = 1$	0.06814	25.60	0.7727	6.60	6.04	$\mathbf{+0.42}$	$\mathbf{+0.0087}$	-0.39
	WaLLoC	$q = 8$	0.06279	24.02	0.7174	6.12	45.45	-1.05	-0.0421	-0.79
	LiVeAction	$q = 9$	0.07338	24.36	0.7309	6.25	1.72	-1.02	-0.0466	-0.91
	MCUCoder	$q = 1$	0.08542	20.55	0.6900	5.21	24.29	-4.30	-0.0884	-2.03
	FRAPPE	$n = 10$	0.05792	24.81	0.7520	6.75	**300.73**	0.00	0.0000	**0.00**
$[0.1,\ 0.215)$	JPEG	$q = 1$	0.17309	21.48	0.6152	4.56	**543.97**	-4.73	-0.1997	-3.72
	JPEG XL	$q = 5$	0.14307	25.81	0.7859	7.15	3.17	-0.39	-0.0426	-0.89
	AVIF	$q = 15$	0.12245	27.26	0.8471	7.78	4.68	$+1.05$	$+0.0179$	-0.05
	mbt2018	$q = 1$	0.11021	28.12	0.8556	7.73	0.17	$\mathbf{+2.16}$	$\mathbf{+0.0255}$	-0.30
	WaLLoC	$q = 16$	0.11444	25.22	0.7906	7.20	30.37	-0.95	-0.0274	-0.54
	LiVeAction	$q = 16$	0.12398	25.37	0.7955	7.25	1.81	-0.96	-0.0315	-0.69
	MCUCoder	$q = 2$	0.15461	23.69	0.7784	6.70	22.69	-3.54	-0.0793	-1.85
	FRAPPE	$n = 13$	0.19661	27.40	0.8740	8.91	167.76	0.00	0.0000	**0.00**
$[0.215,\ 0.464)$	JPEG	$q = 10$	0.32659	26.67	0.8419	7.74	**529.90**	-2.77	-0.0869	-2.40
	JPEG XL	$q = 20$	0.29262	28.70	0.8874	9.64	3.63	$+0.02$	-0.0159	$+0.10$
	AVIF	$q = 35$	0.30673	30.40	0.9303	10.88	4.01	$+1.49$	$+0.0202$	$\mathbf{+0.89}$
	mbt2018	$q = 3$	0.28821	31.39	0.9301	10.05	0.17	$\mathbf{+2.95}$	$\mathbf{+0.0244}$	$+0.46$
	WaLLoC	$q = 36$	0.23783	26.95	0.8711	9.14	19.63	-1.16	-0.0130	$+0.20$
	LiVeAction	$q = 49$	0.34935	28.15	0.9072	10.30	1.73	-1.13	-0.0115	$+0.08$
	MCUCoder	$q = 5$	0.33768	26.12	0.8633	8.80	19.17	-3.10	-0.0583	-1.37
	FRAPPE	$n = 16$	0.31429	29.22	0.9108	10.03	130.73	0.00	0.0000	0.00
$[0.464,\ 1)$	JPEG	$q = 35$	0.72857	31.01	0.9478	12.73	**479.66**	-0.78	-0.0080	$+0.36$
	JPEG XL	$q = 50$	0.51894	31.31	0.9419	12.56	3.60	$+1.20$	$+0.0105$	$+2.14$
	AVIF	$q = 50$	0.60038	33.39	0.9658	13.73	3.33	$+2.61$	$+0.0257$	$\mathbf{+2.54}$
	mbt2018	$q = 5$	0.63418	35.14	0.9694	12.88	0.16	$\mathbf{+4.15}$	$\mathbf{+0.0280}$	$+1.62$
	WaLLoC	$q = 80$	0.50761	29.60	0.9355	12.00	14.66	-0.72	$+0.0003$	$+1.00$
	LiVeAction	$q = 81$	0.56666	30.13	0.9452	12.46	1.63	-0.48	$+0.0062$	$+1.24$
	MCUCoder	$q = 10$	0.60788	27.53	0.9044	10.57	15.22	-3.28	-0.0395	-0.85
	FRAPPE	$n = 20$	0.72202	31.57	0.9431	11.57	73.60	0.00	0.0000	0.00

Each codec's mean distortion difference vs FRAPPE-Image at matched bpp. Positive BD-PSNR / BD-SSIM / BD-DISTS means the test codec achieves higher quality than FRAPPE in that rate regime.

TABLE II
PSNR-BINNED BD-RATE ON KODAK

Regime [PSNR dB]	Codec	Setting	bpp	PSNR (dB)	SSIM	DISTS (dB)	Thr. (MPx/s)	BD-Rate$_{\mathrm{PSNR}}$ (%)	BD-Rate$_{\mathrm{SSIM}}$ (%)	BD-Rate$_{\mathrm{DISTS}}$ (%)
< 22.5	JPEG	$q = 1$	0.17309	21.48	0.6152	4.56	543.97	$+1044.7$	$+769.8$	$+943.1$
	WaLLoC	$q = 1$	0.00968	20.43	0.5165	4.05	74.83	$+54.6$	$+59.8$	$+44.8$
	MCUCoder	$q = 1$	0.08542	20.55	0.6900	5.21	24.29	$+760.1$	–	$+296.0$
	FRAPPE	$n = 3$	0.00800	21.08	0.5455	4.17	**914.94**	**0.0**	**0.0**	**0.0**
$[22.5,\ 25)$	JPEG	$q = 5$	0.22115	23.85	0.7326	5.80	**530.25**	$+516.0$	$+489.1$	$+558.1$
	WaLLoC	$q = 4$	0.03244	22.79	0.6373	5.21	66.27	$+55.8$	$+51.4$	$+61.8$
	LiVeAction	$q = 4$	0.03406	22.98	0.6376	5.13	1.90	$+53.9$	$+56.9$	$+75.7$
	MCUCoder	$q = 2$	0.15461	23.69	0.7784	6.70	22.69	$+361.2$	$+129.2$	$+225.0$
	FRAPPE	$n = 8$	0.04101	24.05	0.7159	6.24	415.67	**0.0**	**0.0**	**0.0**
$[25,\ 27.5)$	JPEG	$q = 10$	0.32659	26.67	0.8419	7.74	**529.90**	$+165.0$	$+168.0$	$+196.9$
	JPEG XL	$q = 5$	0.14307	25.81	0.7859	7.15	3.17	$+22.6$	$+50.1$	$+52.4$
	AVIF	$q = 5$	0.07497	25.83	0.7865	6.81	5.78	$\mathbf{-27.4}$	$\mathbf{-12.9}$	$+16.2$
	WaLLoC	$q = 25$	0.17037	26.11	0.8391	8.29	24.64	$+56.8$	$+31.7$	$+31.1$
	LiVeAction	$q = 25$	0.18668	26.31	0.8461	8.35	1.93	$+57.8$	$+35.8$	$+39.4$
	MCUCoder	$q = 6$	0.39355	26.56	0.8717	9.30	18.17	$+220.2$	$+103.3$	$+113.2$
	FRAPPE	$n = 12$	0.08027	25.64	0.7881	7.34	237.35	0.0	0.0	**0.0**
$[27.5,\ 30)$	JPEG	$q = 20$	0.50827	29.14	0.9139	10.25	**469.02**	$+61.4$	$+60.7$	$+47.2$
	JPEG XL	$q = 20$	0.29262	28.70	0.8874	9.64	3.63	$+3.3$	$+24.7$	-0.1
	AVIF	$q = 25$	0.18705	28.59	0.8903	9.08	4.61	-32.6	-22.2	$\mathbf{-19.6}$
	mbt2018	$q = 1$	0.11021	28.12	0.8556	7.73	0.17	$\mathbf{-50.1}$	$\mathbf{-23.0}$	$+5.2$
	WaLLoC	$q = 56$	0.35890	28.29	0.9100	10.71	20.83	$+38.6$	$+15.0$	-9.6
	LiVeAction	$q = 49$	0.34935	28.15	0.9072	10.30	1.73	$+41.6$	$+14.0$	-3.5
	MCUCoder	$q = 11$	0.66242	27.71	0.9090	10.81	14.62	$+203.2$	$+120.4$	$+51.2$
	FRAPPE	$n = 16$	0.31429	29.22	0.9108	10.03	130.73	0.0	0.0	0.0
$[30,\ 32.5)$	JPEG	$q = 40$	0.78557	31.42	0.9530	13.25	**469.90**	$+15.6$	$+4.3$	-18.9
	JPEG XL	$q = 30$	0.39710	30.04	0.9201	11.14	3.65	-22.4	-17.4	-34.9
	AVIF	$q = 40$	0.37911	31.25	0.9439	11.60	3.53	-43.2	-42.0	$\mathbf{-44.2}$
	mbt2018	$q = 3$	0.28821	31.39	0.9301	10.05	0.17	$\mathbf{-57.7}$	$\mathbf{-45.4}$	-21.6
	WaLLoC	$q = 100$	0.61707	30.56	0.9501	13.26	14.19	$+20.8$	-18.1	-44.0
	LiVeAction	$q = 81$	0.56666	30.13	0.9452	12.46	1.63	$+12.1$	-16.9	-38.2
	FRAPPE	$n = 20$	0.72202	31.57	0.9431	11.57	73.60	0.0	0.0	0.0

Each codec's mean percentage rate difference vs FRAPPE-Image at matched distortion. Negative BD-Rate means the test codec needs less rate than FRAPPE to reach the same quality. The PSNR column anchors the regime; SSIM/DISTS BD-Rate cells use the same PSNR-binned slice and may yield "–" where SSIM/DISTS do not overlap despite matched PSNR.