RAS: A Bit-Exact rANS Accelerator For High-Performance Neural Lossless Compression

Yuchao Qin[1,2,3], Anjunyi Fan[1,2], Bonan Yan[1,2,3*]
[1]Institute for Artificial Intelligence, Peking University, Beijing, China
[2]School of Electronic Engineering and Computer Science, EECS, Peking University, Beijing, China
[3]School of Integrated Circuits, Peking University, Beijing, China
Email: bonanyan@pku.edu.cn

***Abstract*—Data centers handle vast volumes of data that require efficient lossless compression, yet emerging probabilistic models based methods are often computationally slow. To address this, we introduce *RAS*, the Range Asymmetric Numeral System Acceleration System, a hardware architecture that integrates the rANS algorithm into a lossless compression pipeline and eliminates key bottlenecks. RAS couples an rANS core with a probabilistic generator, storing distributions in BF16 format and converting them once into a fixed-point domain shared by a unified division/modulo datapath. A two-stage rANS update with byte-level re-normalization reduces logic cost and memory traffic, while a *prediction-guided decoding* path speculatively narrows the cumulative distribution function (CDF) search window and safely falls back to maintain bit-exactness. A multi-lane organization scales throughput and enables fine-grained clock gating for efficient scheduling. On image workloads, our RTL-simulated prototype achieves 121.2× encode and 70.9× decode speedups over a Python rANS baseline, reducing average decoder binary-search steps from 7.00 to 3.15 (approximately 55% fewer). When paired with neural probability models, RAS sustains higher compression ratios than classical codecs and outperforms CPU/GPU rANS implementations, offering a practical approach to fast neural lossless compression.**

***Index Terms*—rANS; probabilistic circuits; entropy coding; multi-lane architecture; hardware–software codesign.**

I. Introduction

The proliferation of data-intensive applications–such as image and video streaming alongside real-time analytics—has intensified the enduring trade-off between compression efficiency and throughput. While lossy compression schemes can reduce bit rates at the expense of distortion [1], [2], many deployments necessitate bit-exact recovery. This drives the adoption of lossless compression pipelines based on entropy coding, including Huffman coding [3], arithmetic coding [4], [5], mix-algorithm [5], [6], standards like JPEG, JPEG-LS, and PNG [7]–[9], WebP [10], as well as modern asymmetric numeral systems (ANS) variants [11]. However, conventional software implementations often fail to achieve both high compression ratios and low latency under real-time and energy constraints in very-large-scale datacenter application scenarios, highlighting a critical area for improvement.

To address these challenges, an emerging strategy integrates integer entropy coding with learned probabilistic models. Specifically, the range variant of asymmetric numeral systems (rANS) [11] is highly amenable to hardware implementation and, when paired with context-conditioned distributions from neural or probabilistic predictors, can achieve near-entropy performance. Recent compressors leveraging large models exemplify this paradigm by incorporating autoregressive probabilities into arithmetic or rANS coders, achieving state-of-the-art lossless compression rates for diverse data types such as text, images, audio, and video [12]–[16]. The hardware-software codesign landscape (encompassing modeling techniques, algorithmic selections, and system integration) is summarized in Fig. 1.

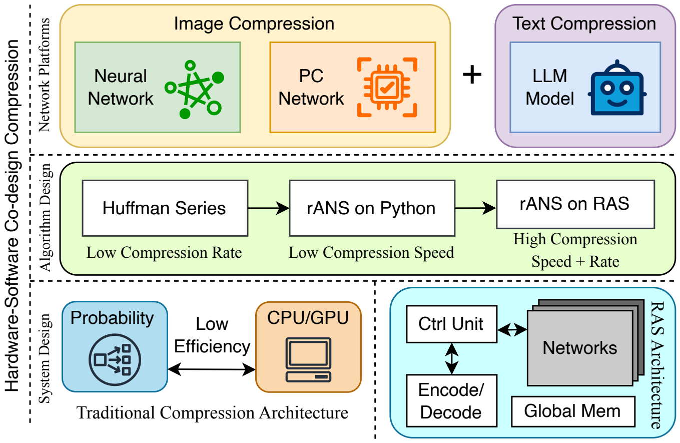

Figure 1. Overview of the hardware-software codesign for learned lossless compression. *Modeling:* neural/PC models for images and LLMs for text provide calibrated probabilities. *Algorithmic:* progression from Huffman and software rANS to **rANS on *RAS*** (this work) increases throughput and compression ratio. *System/Hardware:* RAS integrates a control unit, networks, pipelined encode/decode, and shared global memory–replacing a low-efficiency CPU/GPU + probability generator while preserving bit-exactness.

Building on this codesign framework, this paper introduces *RAS*, an rANS-based accelerator design, aiming to maintain determinism and bit-exact recovery while overcoming the key throughput limitations in learned lossless compression. The architecture incorporates three key circuit and architecture design techniques: (a) a unified mixed-precision division/modulus datapath with byte-level re-normalization to minimize logic and memory overhead; (b) a two-stage rANS update that optimizes the balance between arithmetic operations and normalization for sustained high throughput; and (c) a *prediction-guided decoding* technique that speculatively reduces the cumulative distribution function (CDF) search space with fallback mechanisms to preserve bitstream integrity.

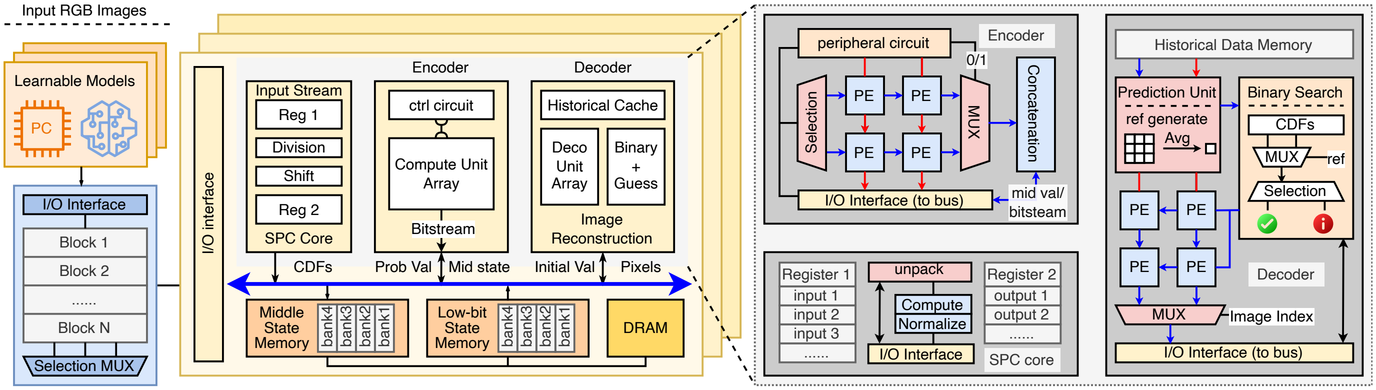

Figure 2. Overall *RAS* architecture. **Left:** Learnable models (orange block) produce absolute distributions that are stored in Global Memory (blue block); the *SPC* performs a single BF16→fixed-point conversion with mass correction and streams shared CDF/frequency tables. **Middle:** The rANS *Encoder* and *Decoder* share a mixed-precision div/mod datapath structure with a two-stage update (parallel quotient/remainder) and byte-level re-normalization; per-lane *Middle-State* and *Low-bit State* memories sustain throughput. **Right:** A prediction-guided decoding path proposes a trial symbol and verifies it against the CDF, falling back on mismatch—reducing average search while preserving bit-exactness. A simple multi-lane fabric with arbitration and clock gating scales throughput without changing the bitstream.

With RTL-based simulation experiments on image workloads, *RAS* achieves 121.2× encode and 70.9× decode speedups over a Python rANS baseline, and reduces average decoder binary-search steps from 7.00 to 3.15 ($\approx 55\%$ fewer).

II. Preliminaries

A. rANS Algorithm Acceleration

rANS approaches Shannon's limit [1] by updating a single integer state using table-indexed operations with power-of-two re-normalization, making it well suited to hardware compare to other entropy coding. In modern compressors [17], [18], rANS is paired with learned priors—e.g., probabilistic circuits (PC) [19] or compact neural networks—that emit context-conditioned symbol distributions subsequently quantized into frequencies or CDFs [19]. Because the rANS pipeline consists of two tightly coupled kernels (state update in encoding and state-to-symbol inversion in decoding), both directions must be accelerated without breaking bit-exactness. Prior work typically optimizes only one side, or omits a hardware–software co-design that simultaneously reduces table-storage pressure, arithmetic cost, and data movement [20]–[22]. In contrast, our approach targets both encode and decode with shared mixed-precision tables, balanced div/mod datapaths, and CDF-aware decoding (Sec. IV), preserving determinism while raising throughput and energy efficiency.

B. Data Compression on Modern Processors

Recent advances aim to improve both compression ratio and speed. To raise throughput, dedicated on-chip compression engines [23]–[25] avoid host overheads and exploit fine-grained parallelism. To raise efficiency, entropy coding has become central in contemporary pipelines, routinely outperforming traditional schemes when supplied with calibrated probabilities. Within these pipelines, learnable probability generators based on traditional neural networks [26], [27] or PCs refine per-symbol distributions that feed a downstream coder. However, few hardware systems tightly couple such predictors with the entropy core while controlling memory footprint and interconnect traffic. Our design explicitly pairs a PC-based (or compact NN) probability module with rANS via a shared fixed-point interface and format-preserving mass correction, enabling higher compression ratios and lower latency without altering the bitstream; implementation details appear in Sec. IV.

III. System Overview

Figure 2 illustrates the RAS datapath and its interfaces. A model-agnostic *neural compression module* (e.g., PC or compact NN) generates *absolute* symbol distributions, which are stored in *Global Memory* using BF16 precision. The *Streaming Prefetch Converter (SPC)* then converts these distributions once into fixed-point CDF or frequency tables, aligned with the rANS radix, and streams them via a backpressure-aware on-chip *bus*. The rANS fabric comprises an *Encoder* and a *Decoder* that share a mixed-precision division and modulo datapath. Each engine incorporates local control logic and *Middle-State (MS) Memory* to sustain line-rate operation while preserving determinism and bit-exactness.

Building on this architecture, the Encoder employs a stationary dataflow where the rANS state and symbols remain resident, while probability entries stream through the shared arithmetic unit. The update process involves a two-stage pipeline: parallel quotient and remainder paths followed by byte-level re-normalization, which maintains high pipeline occupancy and amortizes conversion overhead off the critical path. The Decoder mirrors this structure but focuses on state-to-symbol recovery. To enhance efficiency, a baseline gated binary search over the CDF is accelerated via *prediction-guided decoding* (shown in Fig. 3): a lightweight trial symbol and narrow window are derived from local context, promptly verified against the CDF, and either committed or rolled back

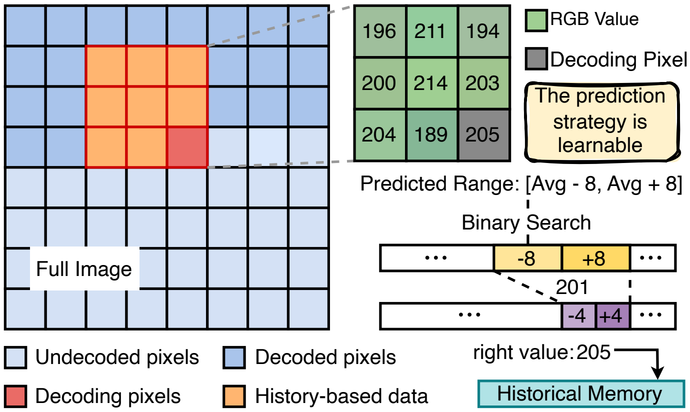

Figure 3. Prediction-guided rANS decoding: the neighborhood average (201) defines a window $[\mathrm{Avg}-8, \mathrm{Avg}+8]$; a dichotomous refinement ($\pm 8 \to \pm 4$) resolves the symbol, yielding the correct value 205.

with bounded cost, reducing average search depth without modifying bitstream semantics.

Furthermore, parallelism is implemented through multiple independent lanes that operate behind the shared *SPC* and *Global Memory* complex. Each lane autonomously issues prefetches, maintains private *MS Memory*, and arbitrates for *BUS* access via a simple credit scheme; stalled lanes relinquish bandwidth to ready lanes, enhancing utilization and enabling lane-level clock gating. Crucially, since probability generation and rANS formatting remain unmodified, the *RAS* architecture ensures full interoperability with existing learned models and software decoders.

IV. Design Techniques

A. Mixed-Precision Probability Module

We propose a mixed-precision processing method for rANS encoding and decoding, which optimizes storage and computational efficiency. Conditional and cumulative probability tables are stored in BF16 format, while the rANS state is retained as a 32-bit unsigned integer. A mixed-precision unit performs a single conversion of BF16 inputs into a shared fixed-point domain, with fractional precision aligned to the re-normalization radix 2^n. Given calibrated probabilities $\{p_x\}$, integer frequencies are computed as $f(x) = \max(1, \mathrm{round}(p_x 2^n))$, followed by a brief mass-correction pass to ensure $\sum_x f(x) = 2^n$ and that the CDF $C(x) = \sum_{y<x} f(y)$ remains strictly monotonic. The CDF is cached for reuse by both encoder and decoder. Subsequent division and remainder operations are executed entirely in the fixed-point domain, avoiding repeated type conversions on the critical computational path. Exponent alignment, incorporating a small guard margin, bounds conversion errors to within one unit in the last place (ULP) of the fixed-point representation. A pipelined divider generates the quotient, while a matched path computes the remainder; both results are power-of-two normalized and forwarded as 32-bit values. Deterministic, monotone rounding (using floor for the quotient) preserves rANS invariants and guarantees bit-exact encoder-decoder consistency across lanes and runs. This design halves table storage compared to single-precision approaches, reduces off-chip bandwidth demands, and simplifies timing closure at high operating frequencies.

B. Encoding Pipeline

Let s_{i-1} be the rANS state before symbol x_i, $f(x_i)$ its quantized frequency, and $C(x_i)$ the CDF value immediately preceding x_i. With re-normalization radix $R = 2^n$, the range-ANS update can be written as

$$s_i = \left\lfloor \frac{s_{i-1}}{f(x_i)} \right\rfloor \cdot R + \big(s_{i-1} \bmod f(x_i)\big) + C(x_i), \tag{1}$$

followed by re-normalization to the canonical unsigned range. We exploit the algebraic separability of the quotient and remainder paths by computing $a_1 = \lfloor s_{i-1}/f(x_i) \rfloor \cdot R$ and $a_2 = (s_{i-1} \bmod f(x_i)) + C(x_i)$ in parallel using the shared fixed-point probabilities, then forming $s_i = a_1 + a_2$ before byte-level re-normalization. re-normalization emits one byte per step while maintaining $s \in [L, RL)$ for a fixed lower bound L, ensuring identical coder/decoder state evolution. Common normalization and exponent handling are performed once and fanned out to both arithmetic paths, reducing redundant logic and balancing pipeline stages. In practice, this two-stage organization improves steady-state throughput (one symbol per cycle after fill) while preserving determinism and exactness.

C. Speculative Prediction for rANS Decoding

Decoder latency is often dominated by interactions with probability tables and the binary search over the CDF required to map the state back to a symbol. Inspired by speculate-and-verify control and branch prediction in CPU [28], [29], we introduce a *decoder-side* prediction path that proposes a trial symbol from lightweight context, advances the rANS state as if the proposal were correct, and immediately verifies this advancement against the reference CDF. On a match, the decoder commits and bypasses the full search; on a mismatch, a short, bounded restore returns to the precise pre-speculation state and the conventional lookup proceeds. Because speculation never alters the bitstream or frequency tables, rANS invariants are maintained and worst-case latency remains identical to the baseline.

The predictor emits an anchor μ and a tolerance Δ, defining a search bracket $[\mu - \Delta, \mu + \Delta]$ over the symbol alphabet. The controller performs a window-gated binary check centered at μ and refines only when necessary, thereby reducing the effective domain from $|\Sigma|$ to $|R| = 2\Delta + 1$. The expected search depth drops from $\log_2 |\Sigma|$ to $\log_2 |R| + \nu$, where $\nu \ll 1$ captures a small amortized verification overhead hidden by the pipeline. If the true symbol lies outside the bracket, the controller expands the window and falls back to the baseline search with a bounded penalty. Unlike encoder-side predictors in PNG/JPEG-LS that reduce residual entropy, our mechanism targets the decoder inner loop to prune CDF work. We deliberately employ hardware-cheap statistics (e.g., neighbor averages with last-value/zero fallback and simple pattern cues) to keep latency fixed and area low while materially reducing

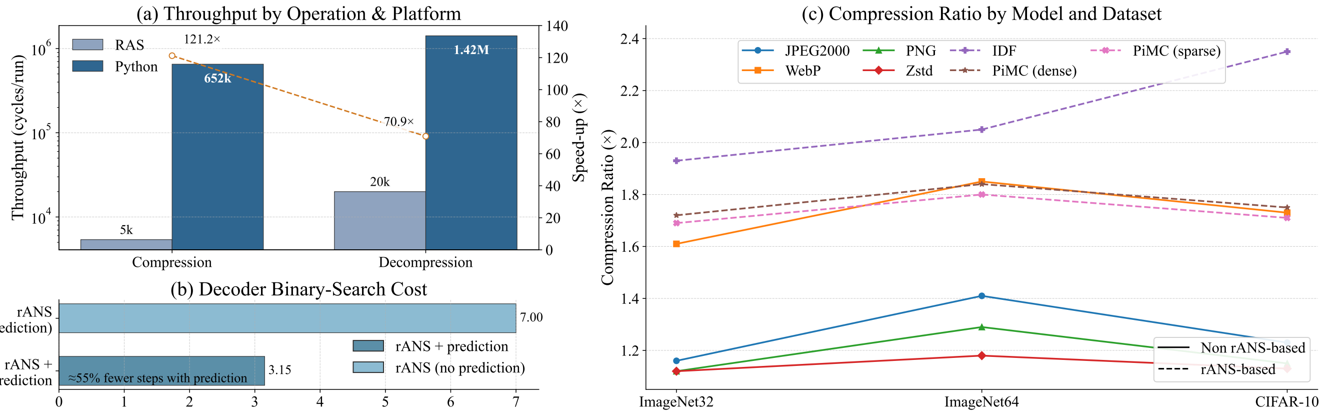

Figure 4. Design exploration results (simulated): (a) Cycle-normalized compute cost (cycles/run; lower is better) for compression and decompression, comparing RAS with a Python rANS baseline; annotations show speed-ups of 121.2× (encode) and 70.9× (decode). (b) Decoder binary-search cost: average steps per symbol drop from 7.00 to 3.15 with prediction (≈ 55% fewer). (c) Compression ratio on ImageNet32/64 and CIFAR-10 for classical codecs (solid) and neural rANS-based models (dashed); neural models with rANS algorithm (IDF, PiMC) achieve higher ratios.

table probes and memory traffic; more expressive fixed-point predictors can be plugged in without changing the interface.

V. Evaluation

Simulated Design Exploration: In this work, we perform comprehensive RTL simulation along two axes: (i) end-to-end encode/decode throughput and (ii) compression ratio on standard image benchmarks. The rANS core is agnostic to the probability generator and interoperates with learned priors, including IDF [30] and PiMC. To isolate the coder, the probability module first produces all required distributions and writes them to shared memory in BF16; the rANS engines then fetch from this cache during encode/decode. The hardware is implemented in SystemVerilog; cycle counts are obtained from RTL simulation (Icarus Verilog). For software comparison, we employ a Python rANS implementation on an Apple M4 and normalize by *cycles* rather than wall time; when a frequency is required, we conservatively take 2.9 GHz [31]. Both systems use the same symbolization and CDFs, so the bitstreams are identical.

System Prototype: With the proposed RAS codesign framework, the lossless compression prototype system is demonstrated at https://vimeo.com/1106157880.

A. Speed

Fig. 4(a) reports cycle-normalized compute cost (lower is better). RAS achieves 121.2× speedup for encoding and 70.9× for decoding over the Python baseline. Measurements include the full coder kernels—division/modulus, CDF access, byte-level re-normalization, state moves, and (for decode) state-to-symbol search and verification—and exclude host I/O, dataset parsing, and probability generation. The gains arise from three effects: (i) the two-stage update that balances arithmetic with re-normalization and sustains one-symbol-per-cycle throughput after pipeline fill; (ii) single-pass BF16→fixed-point conversion with mass-corrected frequencies, which removes repeated casts from the critical path and reduces bus traffic; and (iii) shared tables plus local MS memories, which improve locality and reduce stalls. Decoder throughput is typically constrained by CDF lookups and the state-to-symbol search. As isolated in Fig. 4 (b), prediction-guided decoding lowers the average binary-search depth from 7.00 to 3.15 steps (≈ 55% fewer). Because each step triggers a probability-table access, the reduction directly translates into lower latency and memory traffic while preserving worst-case behavior.

B. Compression Ratio

Compression ratios on ImageNet32/64 [32] and CIFAR-10 [33] are shown in Fig. 4 (c) [1]. We report CR $=$ original bytes/compressed bytes (higher is better). Classical lossless codecs (JPEG2000 [34], WebP, PNG, Zstd; solid curves) improve with increased spatial context but remain below rANS-based neural models (IDF, PiMC dense/sparse; dashed curves) across datasets. RAS is format-preserving: the probability tables, CDF, and symbolization are identical to the software pipeline, and the BF16→fixed-point conversion uses a deterministic mass-correction that enforces $\sum_x f(x) = 2^n$. Consequently, RAS reproduces the exact bitstreams of the reference implementation while accelerating the coding path, so the compression ratios in Fig. 4 (c) transfer unchanged.

VI. Conclusion

We introduce *RAS*, an accelerator that integrates an rANS core with a mixed-precision probability path, a two-stage update pipeline, and a prediction-guided decoder to optimize CDF searches while preserving bit-exactness. The *predict-and-verify* approach generalizes to ANS variants like tANS and iANS, pruning decoder loops without altering bitstreams, and can enhance arithmetic decoding by biasing interval tests toward probable outcomes. Future work will (i) measure PPA on silicon and integrate on-chip probability generators, and (ii) explore lightweight ML predictors [35], [36] that raise accuracy while preserving exactness. Overall, RAS offers a practical, extensible path to high-throughput, bit-exact neural lossless compression using rANS algorithm.

[1] Throughput in MB/s can be derived from processed bytes, cycles, and clock frequency; we use cycles to avoid cross-platform artifacts.

References

[1] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3–4, pp. 379–423, 623–656, 1948.

[2] S. Elakkiya and K. S. Thivya, "Comprehensive review on lossy and lossless compression techniques," *Journal of The Institution of Engineers (India): Series B*, vol. 103, no. 3, pp. 1003–1012, Jun. 2022. [Online]. Available: https://doi.org/10.1007/s40031-021-00686-3

[3] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[4] J.-J. Ding, I.-H. Wang, and H.-Y. Chen, "Improved efficiency on adaptive arithmetic coding for data compression using range-adjusting scheme, increasingly adjusting step, and mutual-learning scheme," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 12, pp. 3412–3423, 2018.

[5] J. Alakuijala, A. Farruggia, P. Ferragina, E. Kliuchnikov, R. Obryk, Z. Szabadka, and L. Vandevenne, "Brotli: A general-purpose data compressor," *ACM Trans. Inf. Syst.*, vol. 37, no. 1, Dec. 2018. [Online]. Available: https://doi.org/10.1145/3231935

[6] Y. Collet and M. Kucherawy, "Zstandard Compression and the 'application/zstd' Media Type," RFC 8878, Feb. 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc8878

[7] G. K. Wallace, "The jpeg still picture compression standard," *Commun. ACM*, vol. 34, no. 4, p. 30–44, Apr. 1991. [Online]. Available: https://doi.org/10.1145/103085.103089

[8] J. Hua, H. Xu, Y. Du, and L. Du, "Improved jpeg lossless compression for compression of intermediate layers in neural networks based on compute-in-memory," *Electronics*, vol. 13, no. 19, 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/19/3872

[9] S.-G. Miaou, F.-S. Ke, and S.-C. Chen, "A lossless compression method for medical image sequences using jpeg-ls and interframe coding," *IEEE Transactions on Information Technology in Biomedicine*, vol. 13, no. 5, pp. 818–821, 2009.

[10] J. Zern, P. Massimino, and J. Alakuijala, "Webp image format," RFC Editor, Request for Comments 9649, Nov. 2024. [Online]. Available: https://www.rfc-editor.org/rfc/rfc9649

[11] J. Duda, "Asymmetric numeral systems as close to capacity low state entropy coders," 11 2013.

[12] Z. Li, C. Huang, X. Wang, H. Hu, C. Wyeth, D. Bu, Q. Yu, W. Gao, X. Liu, and M. Li, "Lossless data compression by large models," *Nature Machine Intelligence*, vol. 7, no. 7, pp. 794–799, May 2025. [Online]. Available: https://doi.org/10.1038/s42256-025-01033-7

[13] J.-M. Chang, J.-J. Ding, and H.-S. Lin, "Adaptive prediction, context modeling, and entropy coding methods for calic lossless image compression," in *2019 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, 2019, pp. 349–352.

[14] S. Yamagiwa and S. Kuwabara, "Autonomous parameter adjustment method for lossless data compression on adaptive stream-based entropy coding," *IEEE Access*, vol. 8, pp. 186 890–186 903, 2020.

[15] D. Ammous, A. Kessentini, N. Khlif, F. Kammoun, and N. Masmoudi, "Evaluation of the improvement in hierarchical lossless video compression," in *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2023, pp. 01–06.

[16] A. Enttsel, A. Marchioni, G. Setti, M. Mangia, and R. Rovatti, "Enhancing anomaly detection with entropy regularization in autoencoder-based lightweight compression," in *2024 IEEE 6th International Conference on AI Circuits and Systems (AICAS)*, 2024, pp. 273–277.

[17] H. Akutsu, T. Naruko, and A. Suzuki, "Gpu-intensive fast entropy coding framework for neural image compression," in *2021 International Conference on Visual Communications and Image Processing (VCIP)*, 2021, pp. 1–5.

[18] W. Tianwen, "Dual attention entropy model for efficient neural image compression," in *2024 21st International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, 2024, pp. 1–5.

[19] A. Liu, S. Mandt, and G. Van den Broeck, "Lossless compression with probabilistic circuits," in *Proceedings of the International Conference on Learning Representations (ICLR)*, apr 2022. [Online]. Available: http://starai.cs.ucla.edu/papers/LiuICLR22.pdf

[20] F. Lin, K. Arunruangsirilert, H. Sun, and J. Katto, "Recoil: Parallel rans decoding with decoder-adaptive scalability," in *Proceedings of the 52nd International Conference on Parallel Processing*, ser. ICPP '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 31–40. [Online]. Available: https://doi.org/10.1145/3605573.3605588

[21] E. Belyaev and K. Liu, "An adaptive binary rans with probability estimation in reverse order," *IEEE Signal Processing Letters*, vol. 30, pp. 1487–1491, 2023.

[22] M. Pastuła, P. Russek, and K. Wiatr, "Low-cost ans encoder for lossless data compression in fpgas," *International Journal of Electronics and Telecommunications*, pp. 219–219, 03 2024.

[23] M. van Beurden and A. Weaver, "Free Lossless Audio Codec (FLAC)," RFC 9639, Dec. 2024. [Online]. Available: https://www.rfc-editor.org/info/rfc9639

[24] C. J. Deepu, X. Zhang, C. H. Heng, and Y. Lian, "A 3-lead ecg-on-chip with qrs detection and lossless compression for wireless sensors," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 12, pp. 1151–1155, 2016.

[25] B. Liu, C.-H. Heng, G. Wang, and Y. Lian, "On-chip data compression scheme for lung eit signal acquisition and recovery," in *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*, 2018, pp. 1–5.

[26] Z. Wang, S. Yin, F. Tu, L. Liu, and S. Wei, "An energy efficient jpeg encoder with neural network based approximation and near-threshold computing," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018, pp. 1–5.

[27] Q. Zhang, C. Chen, S. Yuan, J. Zhang, J. Yuan, H. Huang, Y. Zhang, R. Pan, X. Jiang, J. Zhao, Y. Li, Y. Yin, L. Zhao, G. Wang, and Y. Lian, "Meta: Data compression and event detection grand challenge 2024 with sprsound dataset," *IEEE Data Descriptions*, vol. 1, pp. 122–130, 2024.

[28] A. Seznec and P. Michaud, "A case for (partially) tagged geometric history length branch prediction," *Journal of Instruction-level Parallelism - JILP*, vol. 8, 02 2006.

[29] T. A. Khan, M. Ugur, K. Nathella, D. Sunwoo, H. Litz, D. A. Jiménez, and B. Kasikci, "Whisper: Profile-guided branch misprediction elimination for data center applications," in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2022, pp. 19–34.

[30] E. Hoogeboom, J. W. Peters, R. van den Berg, and M. Welling, *Integer discrete flows and lossless compression*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[31] Notebookcheck, "Apple m4 (10 cores) processor – benchmarks and specs," https://www.notebookcheck.net/Apple-M4-10-cores-Processor-Benchmarks-and-Specs.835975.0.html, 2024, accessed: 2025-10-07.

[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[33] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep. TR-2009, 2009, accessed: 2025-10-07. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[34] D. Taubman and M. Marcellin, "Jpeg2000: standard for interactive imaging," *Proceedings of the IEEE*, vol. 90, no. 8, pp. 1336–1357, 2002.

[35] S. Zangeneh, S. Pruett, S. Lym, and Y. N. Patt, "Branchnet: A convolutional neural network to predict hard-to-predict branches," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2020, pp. 118–130.

[36] E. Garza, S. Mirbagher-Ajorpaz, T. A. Khan, and D. A. Jiménez, "Bit-level perceptron prediction for indirect branches," in *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*, 2019, pp. 27–38.