International Journal of Computer Applications (0975 - 8887)
Volume 187 - No.20, June 2026

Practical Limits of Lossless Compression for bf16 Transformer LLM Weights, with Companion Measurements on Q4_K Tensors in GGUF Q4_K_M Files

Nimo Rotem
AlphaBell Inc.
ORCID: 0009-0008-0920-2764
Nimo@AlphaBell.com

Ariel Rotem
Independent Contributor

ABSTRACT

This paper quantifies how far bf16 transformer large language model (LLM) weights—and Q4_K-typed tensors inside GGUF Q4_K_M files—can be compressed with no loss of information. Every method is scored under one accounting discipline: a trained profile is billed to its method on each file, each run is checked for a byte-exact roundtrip, and the train/test partition is taken over source models rather than individual tensors. For bf16 weights the marginal-byte ceiling sits at $\approx 1.495\times$ (model-level 95 % confidence interval $[1.487, 1.502]$); the strongest method that could be run end-to-end attains $1.499\times$, while the `bf16_split` coder introduced here reaches $1.488\times$. For Q4_K-typed tensors the marginal-byte ceiling is $\approx 1.076\times$ at the tensor-stream level, dropping to 1.041–$1.045\times$ once whole GGUF artifacts are scored, because Q4_K_M files also carry Q6_K tensors that barely compress (near $1.01\times$); the mixture-CDF coder presented here attains $1.052\times$, whereas a dictionary-trained zstd ends up enlarging the stream once its dictionary is billed per file. Between adjacent same-role weight matrices at bf16 precision no exploitable linear redundancy is found (median Pearson $+0.0004$ over 250 layer pairs drawn from two Qwen2.5 models). Across $11{,}942$ verified-roundtrip method-evaluations spanning $7{,}960$ distinct benchmark rows, not a single roundtrip failed; the contribution is the comparison protocol itself rather than another compressor. Each ceiling is a compression-*ratio* bound—hardware independent and confirmed on weights up to 7B parameters (Qwen2.5-7B-Instruct); the single-core x86 (Sapphire Rapids) host constrains only the throughput figures, which are reported for comparability rather than as deployment numbers.

General Terms

Measurement, Performance, Experimentation, Algorithms

Keywords

Lossless weight compression, bf16 transformer LLMs, entropy ceiling, reproducible benchmarking protocol, GGUF Q4_K quantization, source coding

1. INTRODUCTION

Trained model weights are bulky: in bf16, a 7B-parameter transformer occupies ≈ 14 GB. Replicated across model registries, inference fleets, edge devices, and content-delivery networks, even a $1.5\times$ saving translates into real storage cost. A number of recent papers advertise "lossless" compression of such weights—DFloat11 [1], Cloudflare Unweight [2], ZipNN [3], NeuZip [4], OpenZL [5], and ECF8 [6]—each quoting ratios in the 1.30–$1.50\times$ range on bf16. The question addressed here is how much of that range reflects genuine compressibility and how much reflects accounting choices.

1.1 What "lossless" means in this paper

The word is overloaded. Many recent compression papers (across both LLM quantization and pruning literatures) call a method "lossless" if it preserves *task accuracy* on some benchmark within a tolerance, while the underlying weights are mutated. That is a useful notion, but it is not the one studied here. This paper follows DFloat11 in studying *bit-lossless* compression: the original weight tensor is recoverable byte-for-byte from the compressed bytes, verified with a sha256 equality check on every file in every run. This is the notion relevant when the compressed artifact is what a registry actually ships and the deployed inference server expects identical weights. The two notions of losslessness do not nest cleanly, and conflating them inflates apparent compressibility.

1.2 Motivation

Two observations motivate this work. First, the published bf16 numbers above are not directly comparable. Some count trained dictionaries against per-file ratios; others amortize over a corpus or omit the cost. Some sample a few files for roundtrip verification; others assert it on every byte. Some split train/test by file (so weights from the same model can land on both sides); others split by source model. Reported decimals differ by accounting almost as much as by method. Second, multiple independent methods are converging on a narrow band, but no paper has characterized the *ceiling* implicit in the coder class they share. If the methods cluster at a number, the number is worth knowing.

Two things are done in response. First, the byte-marginal ceiling for bf16 weights is derived (a one-line Shannon argument at measured

byte entropies) and measured across a held-out test set: $\approx 1.495\times$ byte-weighted, model-level confidence interval $[1.487, 1.502]$. Second, a single *comparison protocol*, defined below, is applied uniformly, so that methods satisfying it can be placed in one table without their published-decimal asymmetries.

1.3 The comparison protocol

The comparison protocol is a set of accounting rules a benchmark applies uniformly to every method:

(1) *Profile bytes count against the method on every file.* If a method ships a trained dictionary, cumulative distribution function (CDF), or other side data, its size is added to the compressed payload on every file, not amortized over the corpus. The reason is per-file commutability: a per-file ratio under amortization depends on what else is in the corpus, while a per-file ratio under per-file accounting does not, so two papers reporting per-file ratios on different corpora can still be compared. The per-file convention is also pessimistic by design; in production deployments where one profile is shipped once per registry, the per-shard or per-model accounting (Section 5.3, Table 8) is the practitioner number and is reported separately.

(2) *Every (file, method) pair must satisfy* $\mathrm{sha256}(\text{input}) = \mathrm{sha256}(\text{decompress}(\text{compress}(\text{input})))$. A method that fails this is reported failed, not "lossy with caveats."

(3) *Train/test split is by source model, not by file.* Splitting by file lets a trained method see one weight tensor from a model in training and a sibling tensor from the same model in test, which inflates ratios by 0.005–$0.012\times$ (Table 5 in Section 4) and reverses recommendations on borderline methods.

(4) *Throughput is reported single-core CPU on a fixed reference host* (`c3-highcpu-88`, Intel Sapphire Rapids), separated by compress versus decompress. Multi-core and GPU-resident decoders are policy-relevant for production but not comparable across the methods measured here: DFloat11 and Cloudflare Unweight ship CUDA Hopper kernels, OpenZL ships a CPU toolchain, ZipNN ships a multi-codec CPU library. Throughput numbers that mix kernels written for different hardware classes do not compare; a single CPU axis is therefore used, on which every method that is run has comparable measurement, and the hardware-axis gap to GPU kernels is reported honestly rather than papered over (Section 7).

The cost of rule (1) is that some methods look worse than their own papers report (the dictionary that amortized to nearly zero on a million-file corpus is fully visible on each file here). The cost of rule (4) is that strict-accounting decimals cannot be placed next to DFloat11 and Cloudflare Unweight, whose official kernels need graphics processing units (GPUs); this is discussed honestly rather than filled with un-verifiable numbers (Section 3.3).

1.4 Contributions

The contributions of this paper are four:

(1) A measured byte-marginal ceiling for bf16 weights of $\approx 1.495\times$, derived from byte entropies, validated through 7B (Qwen2.5-7B-Instruct), and consistent with the α-stable theory of stochastic-gradient-descent (SGD)-trained networks invoked in ECF8.

(2) A measured tensor-stream ceiling for Q4 K of $\approx 1.076\times$ with a mechanism (optimized per-block post-training-quantization (PTQ) scaling produces near-uniform nibble distributions in the tested corpus) and a deployable artifact-level number of 1.041–$1.045\times$.

(3) A measured null on adjacent-layer scalar-affine residual schemes at bf16 precision in two Qwen2.5 models, reported as scoped evidence on the simplest such scheme rather than as a general claim about cross-layer compression.

(4) The comparison protocol itself, applied to a vendored benchmark harness and result table that any third party can run on a new method.

Table 1 consolidates the four headline measurements and the sections in which each is established. This paper does not propose a new compressor as the contribution. Two methods (`bf16_split` and a Q4_K block mixture-CDF coder) were built only to test whether the measured ceilings are reachable in practice; both sit at the ceiling rather than beating it.

1.5 Scope

All numbers are for transformer LLM weights ($\leq$ 7B parameters, validated through Qwen2.5-7B-Instruct), in bf16 or Q4_K-in-Q4_K_M, on CPU. The study does not measure AWQ Q4, GPTQ Q4, Q5_K_M / Q6_K / Q8_0, FP8, fp32, INT8, optimizer state, gradients, or activations. fp16 is included as a brief companion in Appendix B. The "practical limits" of the title refer to the compression-*ratio* ceilings, which are set by the statistical structure of trained weights rather than by hardware; they are measured rather than asymptotic, and are reported as validated within this regime, not as extrapolations to larger models, other quantization formats, or non-transformer architectures. The single-core x86 (Sapphire Rapids) restriction applies only to the throughput measurements (Section 3.4); the ratio ceilings are hardware-independent.

2. RELATED WORK

2.1 General-purpose and scientific codecs

General-purpose codecs (gzip, brotli, zstd, xz, bzip2) reach ≈ 1.30–$1.45\times$ on bf16 weights but do not exploit the bit structure of floating point. Scientific floating-point compressors (zfp [7], fpzip [8], SZ3 [9]) assume spatial smoothness across adjacent array entries, a property of simulation grids, not of transformer weight matrices; they are therefore not benchmarked here.

2.2 Format-aware weight codecs

DFloat11 [1] Huffman-codes the bf16 exponent with a hierarchical look-up-table decomposition targeting GPU static-RAM (SRAM) decode; it reports whole-model ratios ≈ 1.47–$1.48\times$ across 9 LLMs. *Cloudflare Unweight* [2, 10] Huffman-codes the bf16 exponent over a per-tensor 16-value palette and falls back to verbatim row storage for rare-exponent rows; applied to multilayer-perceptron (MLP) projections only, it reports $\approx 1.43\times$ on MLP weights and 1.15–$1.28\times$ whole-model depending on bundle scope. *ZipNN* [3] performs per-chunk adaptive codec selection across Huffman/zstd/LZ4/snappy and reports $\approx 1.49\times$ on bf16. *NeuZip* [4] applies entropy coding to bf16 streams paired with GPU primitives. *OpenZL* [5] is a trained-graph framework with a universal decoder; its trained `le-u16` profile is the strongest published reference that can be run under the present protocol

Table 1. Summary of the headline measurements. Ratios are byte-weighted unless noted; intervals are model-level 95 % bootstrap confidence intervals.

Quantity	Unit	Measured value	95 % CI	Established in
bf16 marginal-byte ceiling	tensor-stream	$1.495\times$	$[1.487, 1.502]$	Sec. 4.1
strongest runnable method (OpenZL)	tensor-stream	$1.499\times$	—	Sec. 4.2
`bf16_split` (this paper)	tensor-stream	$1.488\times$	—	Sec. 4.2
Q4_K marginal/joint ceiling	tensor-stream	$1.076\times$	$[1.066, 1.087]$	Sec. 5.1
mixture-CDF $K{=}4$ (this paper)	tensor-stream	$1.052\times$	—	Sec. 5.2
Q4_K_M deployable ratio	artifact	1.041–$1.045\times$	—	Sec. 5.3
Adjacent-layer linear redundancy	250 pairs	Pearson $+0.0004$	$[-0.002, +0.003]$	Sec. 6
Verified roundtrips (zero failures)	method-evals	11,942	—	Sec. 3.2

because the official toolchain is CPU. *ECF8* [6] derives entropy bounds for FP8 lossless compression from the α-stable distribution analysis of SGD-trained networks; this is the theoretical anchor for *why* the bf16 exponent has low entropy. Table 2 summarizes these methods, their reported numbers, their compression scope, the hardware their decoders target, and whether each could be executed end-to-end under the protocol of Section 3.

2.3 The shared empirical fact, and the gaps addressed

The key empirical fact across this literature is that the bf16 exponent in trained transformer weights has entropy ≈ 2.6 bits, far below its 8-bit allocation, reported independently by DFloat11 and Cloudflare Unweight, and consistent with the α-stable prediction $H(\text{exponent}) \in [2.4, 2.9]$ bits. The atlas of Section 4.1 measures byte-`high` entropy at 2.74 bits (the $+0.14$ versus 2.6 is the residual entropy of the sign bit, which byte-`high` includes). Three gaps in the prior work are addressed here: (1) no uniform-accounting comparison among methods that one can run end-to-end; (2) no explicit ceiling proposition for the coder class in use; and (3) no measured null on cross-layer schemes at deployment precision.

3. COMPARISON PROTOCOL AND CORPUS

3.1 Protocol

The four rules of the comparison protocol are stated in Section 1.3. Three operational details complete them.

Aggregation. Every ratio table reports both the geometric mean of per-tensor (or per-file) ratios and the byte-weighted corpus ratio. Headline numbers in the abstract are byte-weighted.

Uncertainty. Model-level bootstrap with 1000 resamples is primary (the conservative confidence interval); tensor-level bootstrap is reported as secondary and is too tight, because tensors within one model are not independent. Leave-one-model-out replicates for the bf16 headline are in Appendix C.

Failed runs. These are recorded and surfaced in the public results JSON-Lines file with no retry.

3.2 Corpus

Eight bf16/fp16 source models (gpt2, distilbert-base-uncased, opt-125m, MiniLM-L6-v2, Qwen2.5-0.5B, TinyLlama-1.1B, Qwen2.5-1.5B for the cross-layer null in Section 6, and Qwen2.5-7B-Instruct as 7B validation) and five Q4_K_M GGUF source models (bartowski builds of Qwen2.5-0.5B/1.5B/7B-Instruct, Llama-3.2-3B-Instruct, and Mistral-7B-Instruct-v0.3) are split by model into train and test. The bf16 test set is 290 tensors ≥ 1 MiB across 3 held-out models; 7B validation is 196 bf16 tensors on Qwen2.5-7B-Instruct. The Q4_K test set is 530 tensors across 3 held-out models. Full pinned revisions (40-character SHAs as actually checked out, with a `prefix_match` field) are in `manifest/model-manifest.json` and Appendix A.

Across the full benchmark there are 11,942 verified roundtrip method-evaluations on 7,960 unique benchmark rows, with zero roundtrip failures.

Results are reported at three units, always labeled in tables: tensor-stream (per-tensor bytes; the unit for ceiling derivations), shard (tensors from one model file concatenated; the unit for "ship the weights"), and artifact (full `.safetensors` / `.bin` / `.gguf` files including container metadata; the unit for deployment recommendations). Q4_K_M files mix Q4_K and Q6_K tensors per the llama.cpp convention (value projections and feed-forward-down are Q6_K); Q4_K-typed tensors alone are analyzed in Section 5 and Q6_K is included in artifact accounting in Section 5.3.

3.3 What could not be run, and why

The official kernels for DFloat11 (`LeanModels/DFloat11`) and Cloudflare Unweight (`cloudflareresearch/unweight-kernels`) are CUDA GPU kernels (the Cloudflare kernel documentation explicitly targets NVIDIA Hopper H100/H200; DFloat11 requires CUDA 12+). Both are incompatible with the `c3-highcpu-88` reference host. The ZipNN CPU release (`zipnn/zipnn`) installs and runs but fails the byte-exact roundtrip assertion in its default configuration. Rather than ship un-verifiable strict-accounting decimals for any of these, they are discussed as published references and explicitly not placed in Table 5 alongside methods measured end-to-end (see Table 2). Each method's own published bf16 number sits in the band 1.15–$1.49\times$ (with the spread dominated by compression scope—Cloudflare Unweight is MLP-only—and accounting convention rather than coder class), consistent with the $\approx 1.495\times$ marginal-byte ceiling derived below.

3.4 Justifying the protocol's two strictest choices

Two of the protocol's choices warrant explicit defense because they are the strictest available and they directly drive several of the results.

3.4.1 Why per-file profile accounting. The strict alternative is amortization, in which a profile's cost is divided over an assumed corpus size. Per-file accounting is chosen because (a) it produces decimals that are comparable across papers without requiring authors to agree on a corpus, (b) it makes the profile cost visible at the smallest deployment unit (one weight tensor), and (c) it is

Table 2. Published format-aware weight codecs. "Runnable here?" indicates whether the official implementation could be executed end-to-end with byte-exact roundtrip verification on the CPU reference host. Reported ratios are as stated by each source; scope and decoder hardware explain most of the spread.

Method	Reported bf16 ratio	Compression scope	Decoder hardware	Runnable here?
DFloat11 [1]	1.47–1.48× (whole model)	exponent, all weights	CUDA (Hopper)	No (GPU kernel)
Cloudflare Unweight [2, 10]	1.43× (MLP); 1.15–1.28× whole	exponent, MLP only	CUDA (Hopper H100/H200)	No (GPU kernel)
ZipNN [3]	≈1.49×	adaptive per-chunk	CPU (multi-codec)	Installs; fails byte-exact roundtrip
NeuZip [4]	not directly comparable	entropy + GPU primitives	GPU	No (GPU primitives)
OpenZL `le-u16` [5]	1.499× (this study)	trained graph, byte planes	CPU	Yes
ECF8 [6]	FP8 bound (theory)	exponent (FP8)	— (analysis)	N/A (theoretical anchor)

Table 3. Sensitivity of conclusions to the two strictest protocol choices. Relaxing either choice moves trained-method ratios upward and can reverse a practitioner recommendation.

Protocol choice relaxed	Effect on ratio	Consequence
Model-level → file-level split	+0.005 to +0.012×	inflates trained methods
Per-file → amortized profile	dict-zstd 0.998 → 1.003×	flips "expands" to "compresses"

the case where the trained-versus-untrained distinction is sharpest: small profiles (the 617-byte OpenZL `le-u16`) survive per-file accounting cleanly while large dictionaries (the 112,640-byte dictionary-trained zstd for Q4_K) flip from "compressed" to "expanded." Production deployments amortize one profile across an entire model registry, in which case the per-shard or per-model decimals in Table 8 (Section 5.3) are the practitioner number; these are reported separately rather than as the headline. Table 3 quantifies the sensitivity of the conclusions to these two choices: weakening the train/test split from model-level to file-level inflates trained-method ratios by 0.005–0.012× and flips dictionary-trained zstd on Q4_K from 0.998× (expands) to 1.003× (marginally net positive). The strict accounting is exactly what makes that flip visible.

3.4.2 Why single-core CPU throughput. The restriction is acknowledged as artificial relative to production. It is adopted because the methods that can be placed in a single table use very different runtimes: OpenZL is CPU-native, zstd / brotli / xz are CPU-native, and DFloat11 and Cloudflare Unweight are CUDA kernels. There is no honest way to put a Hopper-kernel decompression number in the same column as an Intel-CPU decompression number; the most defensible choice is to pin one comparable axis, report on it consistently, and then note the GPU-axis gap honestly (Section 7). Multi-core CPU is in principle comparable but is a function of process pinning, non-uniform memory access (NUMA) layout, and parallel batching scheme that drift between releases of the same software; the single-core figure is the one that reproduces. It is reported as a *comparability* number, not a *deployment* number; the latter requires the practitioner's own hardware.

4. THE BF16 COMPRESSION CEILING UNDER MARGINAL-BYTE CODING

4.1 Ceiling and atlas

For a stream of 16-bit values stored as two byte streams $(X_{\text{high}}, X_{\text{low}})$, any coder that codes each stream independently under an independent and identically distributed (iid) marginal model satisfies, by Shannon [11],

$$\text{ratio} \;\le\; \frac{16}{H(X_{\text{high}}) + H(X_{\text{low}})}. \tag{1}$$

Equation (1) is noted as Proposition 1 and not belabored: it is a direct application of the source-coding theorem. Its only subtlety is scope. It does *not* bound coders that use cross-element context within a byte plane (which is how OpenZL slightly exceeds the marginal number below), nor coders using joint two-byte models, nor sub-byte (bit-plane) decompositions. Methods outside this coder class can exceed the marginal number; doing so is not a bound violation.

Table 4 reports the byte-marginal atlas measured across the 290 bf16 test tensors. The exponent-bearing byte (`byte_high`) carries only 2.74 bits of entropy on average, while the mantissa-bearing byte (`byte_low`) sits at 7.97 bits, within 0.03 bits of its 8-bit maximum. The compressible structure is therefore concentrated almost entirely in one of the two byte planes, which is the arithmetic basis for the 1.495× ceiling: the 16-bit value is coded in $2.74 + 7.97 = 10.71$ bits on average.

Figure 1 decomposes these byte-component entropies tensor by tensor: the sign and exponent bits carry almost all of the compressible structure, while the seven mantissa bits sit at their 8-bit ceiling. The flatness of the decomposition across embedding, attention, MLP, and language-model-head categories is itself informative—the compressible structure is a property of trained bf16 weights in general, not of any one layer role. The α-stable theoretical prediction [6], extended from FP8 to the bf16 exponent layout, gives $H(\text{exponent}) \in [2.4, 2.9]$ bits for typical SGD-trained transformers; the measured median 2.74 sits in the middle of that interval, an independent cross-check between theory and measurement.

4.2 Cross-method comparison

Every method that can be run end-to-end under the protocol of Section 3 is placed on the 290-tensor test set in Table 5: general-purpose codecs, the `bf16_split` predictor introduced here, and the official trained OpenZL `le-u16` profile. The three external format-aware methods (DFloat11, Unweight, ZipNN) are intentionally not in this table, for the reasons in Section 3.3 and Table 2.

Figure 2 plots each bf16 test tensor's achieved `bf16_split` ratio against its marginal-byte bound; every point sits just under the $y = x$ ceiling line, the visual statement of the bf16 ceiling. Five observations follow.

(1) General-purpose codecs cluster at 1.36–1.45×, well below the marginal ceiling, because they do not separate the two byte planes.

Table 4. bf16 atlas: byte-marginal ceiling and entropy on the 290-tensor test set (byte-weighted unless noted).

Measure	Median	p10	p90	Model-level 95 % CI
$H(byte\_high)$, bits/byte	**2.74**	2.51	2.93	$[2.62, 2.86]$
$H(byte\_low)$, bits/byte	**7.97**	7.94	7.99	$[7.95, 7.98]$
Distinct values in $byte\_high$	24	19	31	$[21, 28]$
Per-tensor R_{marginal}	$1.498\times$	$1.473\times$	$1.535\times$	$[1.485, 1.501]$
R_{marginal} **byte-weighted (headline)**		$1.495\times$		$[\mathbf{1.487}, \mathbf{1.502}]$

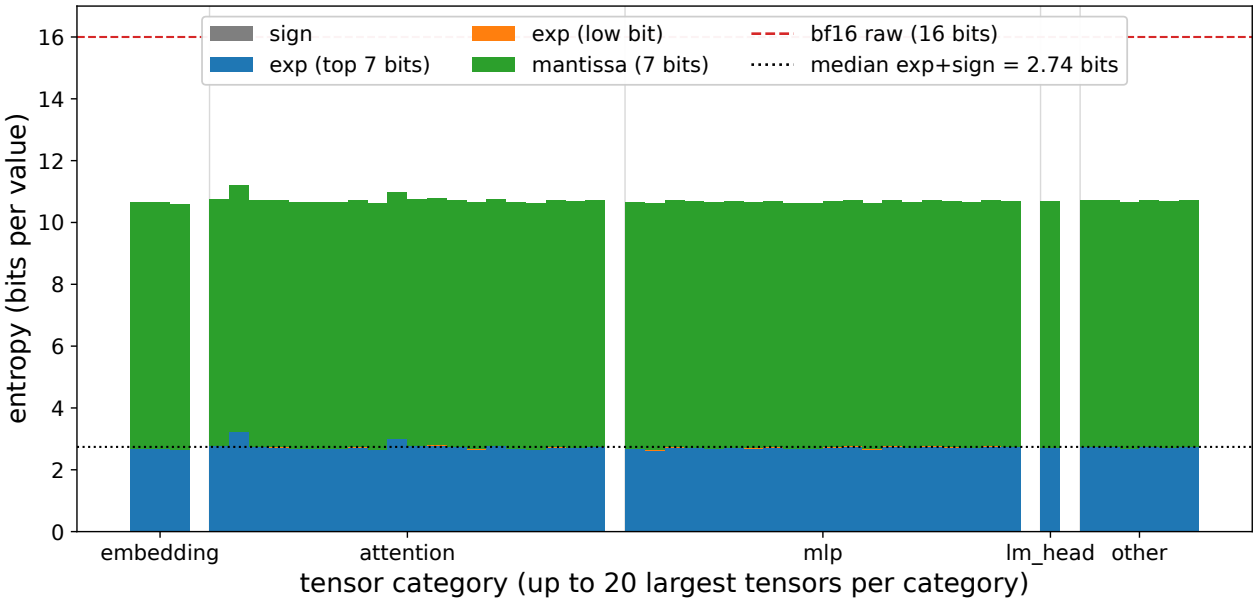

Fig. 1. Per-tensor stacked decomposition of bf16 byte-component entropies (sign, top seven exponent bits, low exponent bit, seven mantissa bits), grouped by tensor category. The mantissa band fills nearly the full 8-bit allocation for every tensor while the sign-and-exponent band stays near 2.7 bits, which is the visual basis for the marginal-byte ceiling.

(2) Byte-grouping the high/low planes captures most of the remaining headroom under a generic entropy coder, pulling zstd-19 from $1.392\times$ to $1.452\times$—a $+0.060\times$ gain from a reversible data layout transform alone, with no change of coder.

(3) The `bf16_split` coder reaches $1.488\times$, $\approx 0.007\times$ below the bound; the gap decomposes into range-asymmetric-numeral-system (rANS) quantization ($\sim 0.005\times$), per-chunk CDF storage ($< 0.001\times$), and chunk metadata ($\sim 0.0002\times$).

(4) Trained OpenZL exceeds the marginal bound by $\approx 0.004\times$ via context coding within byte planes; this matches the measured $H(byte\_high \mid \text{previous } byte\_high)$ of 2.71 bits (0.03 below the marginal), confirming that the excess comes from local context and not from a violation of the bound.

(5) Published external-method ratios in the 1.15–$1.49\times$ range collapse, under uniform accounting on the methods measured here, to 1.358–$1.499\times$; the larger published spread is dominated by compression scope (Unweight is MLP-only) and accounting convention rather than coder class. ZipNN's published $1.49\times$ is within $0.005\times$ of the marginal ceiling, the tightest of the three, supporting but not strictly verifying the framing that byte-plane methods cluster at the marginal-byte ceiling.

4.3 Cross-element structure

Three measurements bound how much can be gained by leaving Proposition 1's coder class. $H(byte\_high \mid \text{previous } byte\_high)$ in row order is 2.71 versus marginal 2.74, i.e. 0.03 bits/byte recoverable, translating to $\approx 0.004\times$ ratio (this is the most plausible mechanism for OpenZL's excess). $H(byte\_high, byte\_low)$ joint is 10.68 versus sum-of-marginals 10.71, i.e. 0.03 bits per 16-bit value, also $\approx 0.004\times$ ratio. Row-shuffle and full-tensor-shuffle do not change $H(byte\_high)$ within 0.005 bits, confirming that the available signal is local rather than long-range. A separate single-element conditional-coding sweep (`bf16_split_ctx`, $K \in \{1, 4, 8, 16, 32\}$) within Proposition 1's class produced no net gain across the K range ($\Delta \leq 0.0003\times$); per-chunk CDF storage cost grew faster than the entropy benefit. The practical reading is that, for bf16 transformer weights, essentially all exploitable redundancy is captured by a good marginal byte-plane model, and the marginal-byte ceiling is therefore the relevant practical limit rather than a loose bound.

Table 5. Cross-method comparison on the bf16 test set, tensor-stream unit. The marginal-byte bound is shown for reference; only trained OpenZL, which uses context coding outside Proposition 1's class, edges above it.

Method	Class	Byte-weighted ratio	Geomean	Decompress MB/s	Profile B
zstd-3	general	1.358×	1.359×	480	0
zstd-19	general	1.392×	1.394×	121	0
zstd-19 + byte-grouping	general (bytegrouped)	1.452×	1.452×	409	0
brotli-q11	general	1.413×	1.415×	38	0
xz-9	general	1.421×	1.424×	8	0
`bf16_split` (this paper)	format-aware	1.488×	1.489×	53	embedded CDF
Trained OpenZL `le-u16`	trained	**1.499**×	**1.499**×	140	617
R_{marginal} byte-weighted bound	—	**1.495**×	—	—	—

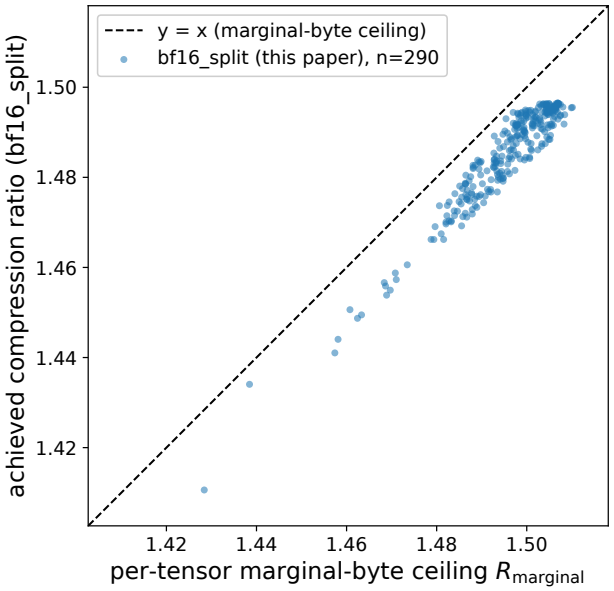

Fig. 2. Per-tensor achieved compression ratio of `bf16_split` versus the per-tensor marginal-byte ceiling R_{marginal} on the 290 bf16 test tensors, with the $y = x$ ceiling line. Every tensor sits just below the line: `bf16_split`, a coder inside Proposition 1's class, reaches but does not exceed the marginal-byte bound. Cross-method numbers (including trained OpenZL, whose byte-weighted ratio edges above the ceiling via context coding) are in Table 5.

4.4 Generalization to 7B

Atlas and `bf16_split` rerun on Qwen2.5-7B-Instruct (held out from all training; 196 bf16 tensors, 12.4 GB) move all paired numbers by less than 0.001×: $H(byte\_high)$ median 2.71 versus 2.74 small-model, R_{marginal} byte-weighted 1.495× versus 1.495× (Δ +0.0001), `bf16_split` 1.488× versus 1.488× (Δ −0.0004), and trained OpenZL 1.499× versus 1.499×. The agreement to the third decimal across a more than ten-fold change in model size is the strongest single piece of evidence that the ceiling is a property of trained bf16 weights rather than of the particular small models used to derive it. The 7B run is a single-model validation, not a parameter-regime sweep; no claim is made or extrapolated above 7B.

5. THE Q4_K COMPRESSION CEILING

All Section 5 results are on Q4_K-typed tensors from GGUF Q4_K_M files. Q6_K tensors are excluded from Sections 5.1–5.2 and included in artifact accounting in Section 5.3. Per `ggml-quants.h`, a Q4_K superblock is 256 elements in 144 bytes (= 4.5 bits/element nominal): 128 bytes of nibble quants, 12 bytes of packed 6-bit sub-block scales and minimums, and 4 bytes of fp16 super-d/$d_{\min}$.

5.1 Atlas and ceiling

Table 6 reports the per-symbol entropies and ceiling measured across the 530-tensor Q4_K test set. The 4-bit nibble alphabet carries 3.86 bits of entropy at the median—only 0.14 bits below the 4.0-bit uniform maximum—which already foreshadows that very little lossless headroom remains once a stream has been quantized to Q4_K.

The headline tensor-stream ceiling is 1.076× (byte-weighted, model-level CI $[1.066, 1.087]$), obtained from a joint within-superblock model in which the nibble stream is coded conditional on its recovered sub-block (scale, minimum) pair. The byte-marginal aggregation that ignores within-superblock conditioning yields 1.044×; the 0.032× gap is the value of within-superblock structure that strict per-byte iid coding cannot exploit. The joint-conditional 1.076× is reported as the Section 5 headline because it is the ceiling for the coder class that the mixture-CDF method actually targets.

Embedding tensors are the only systematic outliers, sitting 0.25–0.30 bits below the corpus median H(nibble); attention and MLP Q4_K tensors are within 0.05 bits of each other across all three test models. The near-uniform finding, visible as the tight mass in Figure 3, is not driven by a single model or layer class.

5.2 Measured methods on Q4_K tensors

Table 7 reports every method run end-to-end on the Q4_K test set under the protocol of Section 3.

Three results are worth flagging.

(1) The mixture-CDF $K = 4$ coder reaches 1.052×, capturing ≈70% of the 1.076× tensor-stream ceiling; the remaining 0.024× is mixture-CDF approximation error plus rANS quantization, analogous to the bf16 gap. $K = 4$ is the sweet spot: Q4_K nibble blocks have so little structure that per-mixture overhead immediately outweighs the marginal modeling gain, and ratios decline monotonically from $K = 4$ to $K = 64$ as the profile grows.

(2) Dictionary-trained zstd-19 *expands* the data on geometric mean under the protocol (0.998×): the 112,640-byte

Table 6. Q4_K atlas: per-symbol entropies and ceiling on the 530-tensor test set.

Measure	Median	p10	p90	Model-level 95 % CI
H(nibble), bits/4-bit symbol	**3.86**	3.71	3.94	$[3.78, 3.91]$
H(byte), bits/byte (packed)	7.73	7.41	7.88	$[7.62, 7.84]$
H(sign), bits/1-bit	0.96	0.91	0.99	$[0.93, 0.99]$
H(magnitude), bits/3-bit	2.91	2.75	2.99	$[2.84, 2.96]$

Table 7. Measured methods on the Q4_K test set, tensor-stream unit, byte-weighted. The dictionary-trained zstd row *expands* the data once its 112,640-byte dictionary is billed per file.

Method	Ratio	Geomean	Decompress MB/s	Profile B
zstd-3	1.034×	1.034×	211	0
zstd-19	1.040×	1.040×	121	0
zstd-19 dict-trained	**0.998**×	0.9985×	124	112,640
xz-9	1.044×	1.044×	10	0
bit-plane decomp + zstd-19	1.043×	—	116	0
mixture-CDF $K = 4$ (this paper)	**1.052**×	1.052×	38	277
mixture-CDF $K = 8$	1.051×	—	35	512
mixture-CDF $K = 16$	1.050×	—	32	712
mixture-CDF $K = 32$	1.049×	—	30	920
mixture-CDF $K = 64$	1.047×	—	28	1,062

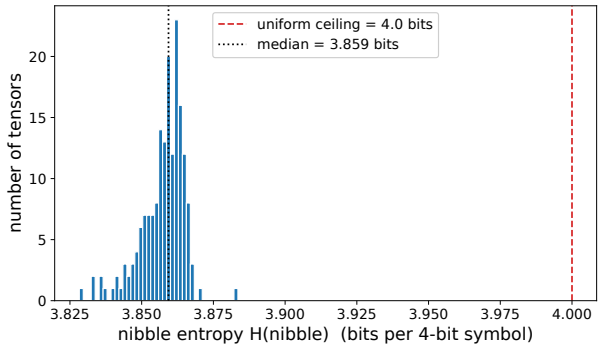

Fig. 3. Distribution of per-tensor nibble entropy H(nibble) over the Q4_K corpus, with the 4.0-bit uniform ceiling and the corpus median annotated. The mass concentrates within a few tenths of a bit of the uniform maximum, which is why no marginal nibble coder can compress Q4_K appreciably.

dictionary charged per file exceeds the per-file gain on a near-uniform stream. Under a file-level random split the same method reads as 1.003× (marginally net positive), reversing the practitioner recommendation; this is the cleanest case for the per-file profile rule.

(3) xz-9 wins a 0.004× ratio edge over zstd-19 but at ≈10 MB/s versus 121 MB/s decompress; zstd-19 sits on the Pareto frontier, xz-9 does not.

The plausible *mechanism* for the near-uniform nibble distribution is that per-block scale optimization during PTQ chooses scales to minimize reconstruction error, and a side effect of a good scale choice is that the 16 quantizer outputs get used roughly evenly; a heavily skewed nibble distribution would imply a poorly-chosen scale. This explanation is offered for the formats measured only; it is not claimed to transfer to AWQ Q4, GPTQ Q4, Q5_K_M, or Q8_0, which were not measured.

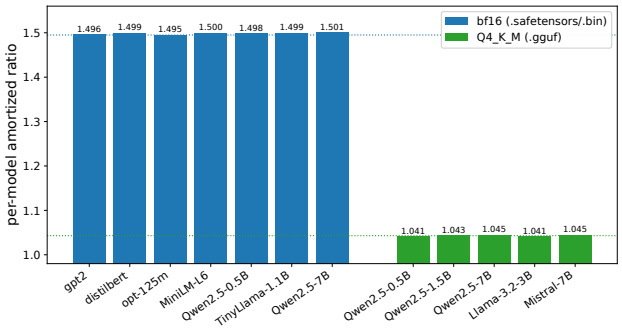

Fig. 4. Per-model amortized artifact ratios from Table 8. bf16 artifacts compress to ≈1.50× regardless of model size, while Q4_K_M artifacts cluster near 1.04× because they are already quantized and dilute the Q4_K gain with near-incompressible Q6_K tensors and container metadata.

5.3 GGUF artifact-level accounting

Q4_K_M files include Q6_K tensors that compress near 1.01×. A five-stream decomposition on the whole GGUF (Q4_K nibbles, packed sub-block scales/minimums, fp16 super-scales, Q6_K passthrough, and metadata) coded with zstd-19, with byte-grouping on the scale headers, reaches 1.0422×; the same artifact-level number is also reached by substituting the mixture-CDF coder for zstd-19 on the nibble stream (1.0416×). Decomposition matters more than the per-stream entropy coder choice: going from whole-file zstd-19 (1.022×) to a five-stream decomposition with zstd-19 on each stream (1.034×) gains +0.012×, while substituting the Q4_K coder on the nibble stream only adds another +0.008×. The practical lesson is that, for already- quantized artifacts, choosing *what to separate* dominates choosing *how to entropy-code* each stream. The artifact ratios across the five GGUF Q4_K_M test files cluster in 1.041–1.045× with median 1.043× (Table 8, Figure 4).

Table 8. Artifact-level results (strongest method per format). The bf16 rows use trained OpenZL `le-u16`; the GGUF rows use the five-stream decomposition described above.

Artifact	Format mix	Input	Per-model amortized ratio
gpt2 (.safetensors)	bf16+fp16	498 MB	1.496×
distilbert-base (.safetensors)	bf16	268 MB	1.499×
opt-125m (.bin)	bf16	500 MB	1.495×
MiniLM-L6-v2 (.safetensors)	bf16	90 MB	1.500×
Qwen2.5-0.5B (.safetensors)	bf16	988 MB	1.498×
TinyLlama-1.1B (.safetensors)	bf16	2.20 GB	1.499×
Qwen2.5-7B (4-shard .safetensors)	bf16	15.20 GB	1.501×
Qwen2.5-0.5B-Q4_K_M (.gguf)	Q4_K+Q6_K+meta	396 MB	1.041×
Qwen2.5-1.5B-Q4_K_M (.gguf)	Q4_K+Q6_K+meta	986 MB	1.043×
Qwen2.5-7B-Q4_K_M (.gguf)	Q4_K+Q6_K+meta	4.68 GB	1.045×
Llama-3.2-3B-Q4_K_M (.gguf)	Q4_K+Q6_K+meta	2.02 GB	1.041×
Mistral-7B-Q4_K_M (.gguf)	Q4_K+Q6_K+meta	4.37 GB	1.045×

The 12-artifact corpus is small. The pattern (per-tensor $\approx$ per-model for bf16 because the 617-byte profile amortizes negligibly; tensor-stream $>$ artifact for Q4_K_M because Q6_K dilutes the gain) is expected from format structure; specific decimals should be read with the small-sample caveat.

6. ADJACENT-LAYER LINEAR RESIDUALS: A MEASURED NULL

6.1 Setup

If transformer layers are statistically similar across depth, an obvious lossless strategy is to store layer K as a small affine transform of layer $K-1$ plus a residual that compresses more than the raw layer. LoRA [12] shows that fine-tuning *updates* are low-rank; layer pruning [13] shows that individual layers can be removed with little accuracy loss; either could motivate cross-layer compression. The simplest version is tested here: scalar-affine residuals between same-role adjacent-layer weight matrices at bf16 precision. For each pair $(K-1, K)$ with the same role (`q_proj`, `k_proj`, `gate_proj`, etc.) and shape, the Pearson correlation, the best scalar affine (a, b), the $H(byte\_high)$ of the residual versus the raw tensor, and the zstd-19 ratio of residual versus raw are computed.

6.2 Results

Across **250 (model, role, K) tuples** drawn from Qwen2.5-0.5B (115 pairs) and Qwen2.5-1.5B (135 pairs), Table 9 summarizes the measured redundancy.

The distribution in Figure 5 sits tightly around zero. The confidence interval rules out $|r| \geq 0.005$ at 95 % confidence; a meaningful scalar-affine residual scheme would need $|r| \gtrsim 0.5$ (entropy reduction is roughly proportional to r^2 for small r). The result is three orders of magnitude below the threshold for usable compression. On the evidence of these two models the direction was halted without building the method, while noting that the null is bounded to the scalar-affine family and the architectures tested.

The cross-layer intuition transfers poorly because LoRA exploits *fine-tuning deltas* (low-rank in the update, not in the absolute weights), pruning exploits *functional importance* (not parameter correlation), and model-surgery results are about *layer outputs* (not parameters). None of these implies that absolute bf16 bit patterns are predictable across adjacent layers.

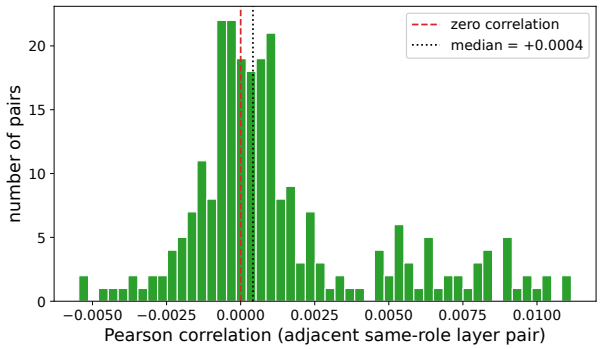

Fig. 5. Distribution of the 250 Pearson correlation values between same-role adjacent-layer weight matrices, with the model-level confidence band overlaid. The mass sits tightly around zero, three orders of magnitude below the $|r| \gtrsim 0.5$ that a usable scalar-affine residual scheme would require.

6.3 Scope of the null

The null is scalar-affine, between adjacent same-role tensors, in two Qwen2.5 models (250 pairs). It is therefore evidence against the *simplest* cross-layer scheme on *this* model family at bf16 precision, not a general result: nonlinear cross-layer schemes, vector-affine or low-rank residuals, cross-checkpoint deltas, and other model families are not tested, and the practitioner recommendation in Section 7 is bounded accordingly. Whether richer cross-layer structure exists in other architectures is an open question that would require its own measurement; the recommendation is against the simple scalar-affine scheme on the evidence presented, not against cross-layer compression in general.

7. DISCUSSION AND PRACTITIONER SUMMARY

7.1 Format-by-strategy recommendations

Table 10 collects the per-format recommendation, expanded by use case below.

Figure 6 places every measured method on the throughput–ratio plane and highlights the Pareto frontier that the recommendations track.

Table 9. Cross-layer scalar-affine residual atlas: 250 same-role pairs across two Qwen2.5 models. All three redundancy measures are statistically indistinguishable from zero.

Measure	Median	p10	p90	Model-level 95 % CI
Pearson correlation	$+\mathbf{0.0004}$	-0.012	$+0.018$	$[-0.002, +0.003]$
$H(byte\_high)$ reduction in residual	-0.001 bits	-0.02	$+0.01$	$[-0.004, +0.002]$
zstd-19 ratio gain (residual / raw)	-0.0002	-0.02	$+0.01$	$[-0.003, +0.002]$

Table 10. Practitioner summary by format and use case.

Format	Recommended strategy	Expected ratio	Decompress MB/s (CPU)
raw bf16	trained OpenZL `le-u16` (617 B profile)	1.498–1.501×	140
raw bf16 (no external profile)	`bf16_split`	1.488×	53
raw bf16 (GPU deployments)	DFloat11 (CUDA; not measured here)	≈1.47–1.48× per [1]	GPU only
GGUF Q4_K_M (artifact)	5-stream decomp + bytegrouped scales + zstd-19	1.041–1.045×	≈110
GGUF Q4_K_M (low-effort)	whole-file zstd-19	1.022×	121
adjacent same-role layers (bf16, Qwen2.5)	do not attempt simple affine schemes	—	—
AWQ Q4 / GPTQ Q4 / Q5_K_M / Q6_K / Q8_0 / FP8 / fp32	not measured	—	—

By use case. For archival storage and CDN distribution of bf16, trained OpenZL is preferred: the $1.5\times$ compounded over a model registry is real and the small profile size amortizes to zero. For cold-load at inference, trained OpenZL again: on 3 GB/s NVMe the disk read dominates and smaller bytes matter more than faster decompression. For edge or on-device deployment where decompress throughput dominates, zstd-3 with byte-grouping is preferred. For GPU-resident inference where SRAM decode matters, DFloat11 has the right architecture; the CPU numbers here do not settle that comparison.

7.2 The hardware-axis gap, stated honestly

DFloat11 and Cloudflare Unweight are designed for GPU SRAM decompression and cannot be placed in the throughput column of this study. Their published bf16 ratios (≈1.47–1.48× for DFloat11; ≈1.43× on MLP for Unweight) sit slightly below the measured OpenZL number under uniform accounting, but the present protocol cannot adjudicate the *throughput* axis on which those kernels are designed to win. A GPU-host extension of the protocol is a clear follow-up.

7.3 What to skip, and what to pursue

What the results say to skip (for the formats measured): dictionary-trained zstd on Q4_K (the dictionary cost exceeds the per-file gain), per-tensor codec routing for bf16 (an oracle router gains $\leq 0.0001\times$ over the single best method), simple scalar-affine cross-layer schemes at deployment precision (on the evidence of two Qwen2.5 models; richer cross-layer schemes and other model families remain untested), and new per-file adaptive codecs aimed at beating the marginal ceiling on bf16 by tuning marginal models alone.

What is worth working on , in order of subjective expected payoff: (a) cross-checkpoint deltas for fine-tuning (the within-model null in Section 6 does not address this; the delta from pretrained to fine-tuned is what LoRA actually exploits); (b) lossless compression of optimizer state (a pre-measurement on one Adam-moment [14] checkpoint reached $1.41\times$ with trained OpenZL, well above the weight ceiling, because the exponential-moving-average aggregation narrows the magnitude band); (c) reversible dtype transforms feeding existing fast codecs; (d) lossless activation compression; and (e) out of scope here but the largest open headroom, *lossy* near-lossless compression bounded by inference-output equivalence rather than bit-equivalence.

8. LIMITATIONS

Format scope. The measured ceilings apply only to the formats measured: bf16 transformer LLM weights $\leq$ 7B (validated through Qwen2.5-7B-Instruct), and Q4_K-typed tensors from GGUF Q4_K_M files. fp16 is reported as a companion in Appendix B and is excluded from the headline contribution count. AWQ Q4, GPTQ Q4, Q5_K_M, Q6_K, Q8_0, FP8, fp32, INT8, optimizer state, gradients, activations, and non-transformer architectures are neither measured nor extrapolated to.

External-method comparison gap. Table 5 omits strict-accounting rows for DFloat11, Unweight, and ZipNN for the reasons in Section 3.3. Closing the gap requires either a Hopper-class GPU benchmark host or a verified-roundtrip ZipNN configuration; both are follow-ups.

Sample sizes. 290 tensors / 3 held-out source models for the bf16 ceiling (the model-level confidence interval is the conservative one quoted in the abstract); 530 tensors / 3 held-out source models for Q4_K; a 12-file artifact-level corpus that is exploratory, where the per-format *pattern* is expected from format structure but the *decimals* should be read with the small-sample caveat; and a 250-pair cross-layer atlas across two Qwen2.5 models, sufficient to rule out large scalar-affine gains but not to address other architectures or nonlinear schemes.

Hardware. Single-core x86 (Sapphire Rapids) only. Ratios are hardware-independent; throughput is CPU-only. ARM and GPU-resident decompression are unmeasured; Section 3.4 explains why the single-core restriction was adopted despite its artificiality.

What the ceiling does and does not prove. Proposition 1 (Eq. 1) bounds only iid-marginal-byte coders on each plane. It does not

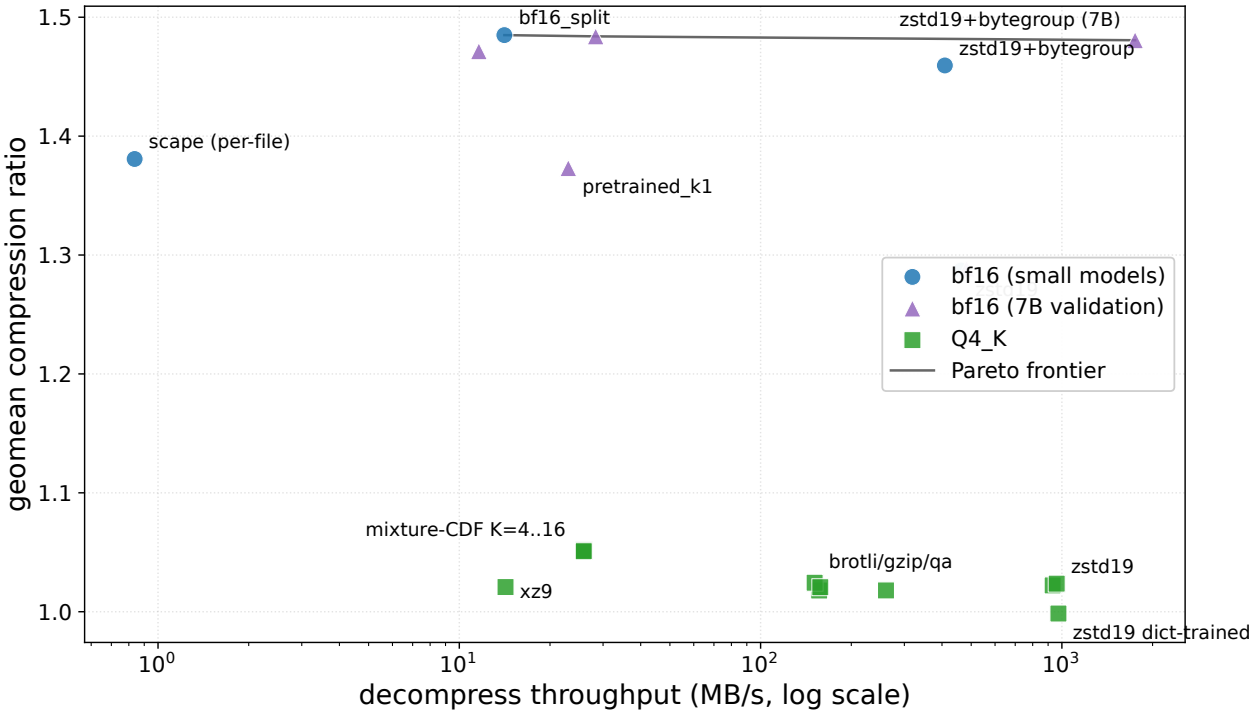

Fig. 6. Decompress throughput (MB/s, log scale) versus byte-weighted compression ratio for every method measured, with the Pareto frontier highlighted. The recommendations in Table 10 track this frontier: trained OpenZL and `bf16_split` dominate the bf16 ratio axis, while zstd-3 with byte-grouping dominates the throughput axis.

bound coders using cross-element context within a byte plane (which is how trained OpenZL exceeds the marginal number by $\approx 0.004\times$), joint two-byte models, or sub-byte decompositions. The "practical ceiling" language is calibrated to acknowledge these limits.

9. CONCLUSION

The measured byte-marginal ceiling for bf16 transformer LLM weights is $\approx 1.495\times$ (model-level CI $[1.487, 1.502]$), validated through 7B, derived from a one-line Shannon argument at measured byte entropies and consistent with the α-stable theory of SGD-trained networks invoked in ECF8. The measured tensor-stream ceiling for Q4_K-typed tensors in GGUF Q4_K_M files is $\approx 1.076\times$, set primarily by the near-uniform nibble distributions that optimized per-block PTQ scaling produces; the deployable artifact-level ratio is 1.041–$1.045\times$ because Q4_K_M files mix in Q6_K tensors that compress near $1.01\times$. No usable linear redundancy is found between adjacent same-role layer weights at bf16 precision in two Qwen2.5 models. The methodological contribution is the *comparison protocol*—strict per-file profile accounting, byte-exact roundtrip verification on every run, model-level train/test split, and single-core CPU throughput on a fixed host—under which dictionary-trained zstd on Q4_K flips from "compresses" to "expands," trained-method ratios under file-level splits inflate by 0.005–$0.012\times$, and the published cross-method spread on bf16 collapses by roughly two-thirds. Both ceilings are compression-ratio limits validated on transformer weights through 7B parameters and are reported as measured within that regime, not as claims about larger models, other quantization formats, or non-transformer architectures; the single-core x86 (Sapphire Rapids) host constrains only the throughput axis, which is reported for comparability rather than as a deployment figure. The boundary is narrow in scope but tight where it applies.

Declaration on the Use of Generative AI

During the preparation of this manuscript the authors used generative AI tools to assist with language editing, figure rendering, and LaTeX typesetting. Generative AI was not used to design the study, to generate or analyze the benchmark data, or to produce any of the reported results, which are entirely the authors' own work. The authors reviewed and edited all AI-assisted material and take full responsibility for the content of this publication.

Acknowledgments

The authors thank Professor Christopher De Sa (Cornell University) for reading the manuscript and providing detailed peer-review-style feedback. The authors also thank the maintainers of OpenZL, llama.cpp, the Hugging Face Hub, and the public model checkpoints used in the corpus for making this measurement reproducible.

10. REFERENCES

[1] Zhang, T., Sui, Y., Zhong, S., Chaudhary, V., Hu, X., and Shrivastava, A. 2025. 70% Size, 100% Accuracy: Lossless LLM Compression for Efficient GPU Inference via

Dynamic-Length Float. In Advances in Neural Information Processing Systems (NeurIPS). arXiv:2504.11651.

[2] Nikulin, I. 2026. Unweight: Lossless MLP Weight Compression for LLM Inference. Tech. Rep. Cf-TR-2026.04.v1, Cloudflare Research.

[3] Hershcovitch, M., Choshen, L., Wood, A., Enriquez, I., Loaiza-Ganem, G., Peleg, T., Kim, H., Klein, T., and Harnik, D. 2024. ZipNN: Lossless Compression for AI Models. arXiv:2411.05239.

[4] Hao, Y., Cao, Y., and Mou, L. 2024. NeuZip: Memory-Efficient Training and Inference with Dynamic Compression of Neural Networks. arXiv:2410.20650.

[5] Collet, Y., Kuzmin, D., Hou, W. F., Mazumder, N., Schlinker, B., Lemire, D., and Lakshman, H. 2025. OpenZL: A Graph-Based Model for Compression. arXiv:2510.03203.

[6] Yang, Z., Zhang, T., Xie, Y., Li, B., Xu, Y., and Shrivastava, A. 2026. To Compress or Not? Pushing the Frontier of Lossless GenAI Model Weights Compression with Exponent Concentration. In International Conference on Learning Representations (ICLR). arXiv:2510.02676.

[7] Lindstrom, P. 2014. Fixed-Rate Compressed Floating-Point Arrays. IEEE Transactions on Visualization and Computer Graphics 20, 12, 2674–2683.

[8] Lindstrom, P., and Isenburg, M. 2006. Fast and Efficient Compression of Floating-Point Data. IEEE Transactions on Visualization and Computer Graphics 12, 5, 1245–1250.

[9] Liang, X., Zhao, K., Di, S., Li, S., Underwood, R., Gok, A. M., Tian, J., Deng, J., Calhoun, J. C., Tao, D., Chen, Z., and Cappello, F. 2023. SZ3: A Modular Framework for Composing Prediction-Based Error-Bounded Lossy Compressors. IEEE Transactions on Big Data 9, 2, 485–498.

[10] Galicer, M., Nikulin, I., and Branch, C. 2026. Unweight: how we compressed an LLM 22% without sacrificing quality. Cloudflare Blog, Apr. 17, 2026.

[11] Shannon, C. E. 1948. A Mathematical Theory of Communication. Bell System Technical Journal 27, 3, 379–423.

[12] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In International Conference on Learning Representations (ICLR). arXiv:2106.09685.

[13] Gromov, A., Tirumala, K., Shapourian, H., Glorioso, P., and Roberts, D. A. 2025. The Unreasonable Ineffectiveness of the Deeper Layers. In International Conference on Learning Representations (ICLR). arXiv:2403.17887.

[14] Kingma, D. P., and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In International Conference on Learning Representations (ICLR). arXiv:1412.6980.

APPENDIX

A. REPRODUCIBILITY

Status. The bf16, Q4_K, and cross-layer empirical core (the $1.495\times$ / $1.076\times$ / 1.041–$1.045\times$ / Pearson $+0.0004$ results) is backed by `results/results.jsonl.zst` (each row tagged with `_stage` / `_kind`; see `results/results_summary.md`). The vendored OpenZL `le-u16` profile (`profiles/bf16.zl`, 617 B) reproduces the OpenZL ratios to within ± 0.001 on the 290-tensor test corpus. Three smoke-test fixtures and their checksums are committed; `reproducibility_smoke_test.sh` runs them in seconds. Eight inference-sanity prompts are committed and end-to-end token identity has been verified on gpt2; a full bf16/fp16 plus Q4_K multi-model sweep is deferred to a GPU host. The figures ship as PDF, PNG, and SVG at `figures/`.

Archive identifiers. The reproduction artifact (code, data, results, profiles, and figures; unchanged for this revision, as no new experiments were run) is at `https://github.com/NimoRotem/llm-compression-limits`, tag `v1.0.0`, with Software Heritage identifier `swh:1:rel:08d597be136278838e8cc2fef2a68303d990208d`. The hosted reproduction-artifact set is at `https://knowva.ai/llm-compression-limits/`.

Pinned corpus. `manifest/model-manifest.json` is authoritative (40-character SHAs as actually checked out, with a `prefix_match` field per model). Of the 13 source-model rows, 7 have prefix mismatches relative to the build-time 8-character prefixes; the manifest captures both.

Smoke test. Table 11 lists the three single-file fixtures and their expected `sha256[:16]` prefixes.

Full results. The per-(file, method) results JSON-Lines file is at `https://knowva.ai/llm-compression-limits/results/results.jsonl.zst` ($7{,}960$ unique benchmark rows / $11{,}942$ verified roundtrip method-evaluations, ~ 0.7 MB compressed). Each row carries `aggregation_unit` $\in$ {tensor_stream, shard, artifact}, `profile_granularity` $\in$ {per_file, per_shard, per_model}, and three Boolean inclusion flags; the analysis notebooks at `notebooks/` recompute each table by filtering on those fields.

End-to-end runtime on `c3-highcpu-88` is ~ 6.5 hours. The build commands are `run_full_benchmark.sh` (corpus in, results out, 88 workers), `summarize.py`, `atlas.py`, `bootstrap_ci.py` (model-level, leave-one-out, seed 42, 1000 resamples), `inference_sanity_check.py`, and `whole_file_gguf_ablation.py`.

B. FP16 COMPANION MEASUREMENT

fp16 is included as a companion measurement only; it is excluded from the headline contribution count for two reasons: (i) fp16 is increasingly being replaced by bf16 in deployment, and (ii) an atlas check found the fp16 byte-marginal headroom below the threshold set for committing to a format-specific method, so no `fp16_split` was developed.

fp16 layout is $S(1)\ E(5)\ F(10)$. Little-endian byte storage gives $byte\_low$ (byte[0]) = the bottom 8 fraction bits (near-uniform) and $byte\_high$ (byte[1]) = sign + 5 exponent bits + top 2 fraction bits. Across 290 fp16 test tensors: $H(byte\_high)$ median 4.41 bits, $H(byte\_low)$ median 7.96 bits, and R_{marginal} byte-weighted $1.293\times$ (CI $[1.275, 1.314]$); the bound is lower than bf16's $1.495\times$ because fp16's $byte\_high$ mixes exponent and fraction bits, raising its entropy.

Trained OpenZL `le-u16` on fp16 reaches $1.470\times$ byte-weighted on the fp16 test corpus, $\approx 0.18\times$ above the fp16 marginal-byte ceiling via the same context-coding mechanism that lifts OpenZL slightly above the marginal number on bf16. The 778 B fp16 profile that produced this result is not vendored in the public artifact set, per the deprioritization decision above; the bf16 profile that is vendored reproduces bf16 ratios within ± 0.001. `bf16_split` applied unmodified to fp16 reaches $1.302\times$, $\approx 0.009\times$ above

Table 11. Reproducibility smoke-test fixtures.

Target	Input fixture	Expected sha256[:16]
`bf16_split`	`TinyLlama_layer3_q_proj.bin` (8 MiB)	`fc3544c35489eb94`
`qb_mixture_k4` (+277 B profile)	`Llama32_3B_layer14_gate.q4k.bin` (13.5 MiB)	`c79972a64d11b717`
`decomp_perstream_zstd19_bg`	`Qwen2.5-0.5B-Q4_K_M.gguf` (379 MiB)	`1c6f077f8c228cf2`

the fp16 marginal ceiling via the same mechanism. The fp16 picture therefore has the same shape as bf16 (marginal-byte ceiling derivable from Proposition 1, trained methods exceeding it via context or bit-plane coding), but with the bound lower ($1.293\times$ versus $1.495\times$).

C. LEAVE-ONE-MODEL-OUT FOR THE BF16 CEILING

For the headline bf16 byte-weighted R_{marginal} over the full test set, Table 12 reports the leave-one-model-out replicates.

No single source model drives the headline. Equivalent leave-one-model-out tables for the Q4_K ceiling and the inter-layer Pearson correlation are at `notebooks/lomo-tables.ipynb`; both show similar robustness.

Table 12. Leave-one-model-out replicates for the bf16 byte-weighted R_{marginal} headline. No single source model drives the headline.

Held-out model	R_{marginal} byte-weighted	Within model-level 95 % CI $[1.487, 1.502]$?
MiniLM held out	$1.498\times$	yes
Qwen2.5-0.5B held out	$1.491\times$	yes
TinyLlama-1.1B held out	$1.496\times$	yes