BiLoCo: Binary Low-Rank Corrections for LLM FP4 Decode

David Jin [*1] **Beshr IslamBouli** [*1] **Tarushii Goel** [1] **Han Guo** [1] **Yoon Kim** [1]

Abstract

Post-training quantization (PTQ) to FP4 has emerged as a key technique for reducing large language model inference cost, especially on Blackwell GPUs which offer native FP4 tensor core support. Direct PTQ to FP4, however, still leaves a noticeable accuracy gap. Recent work on SVDQuant (Li et al., 2025) mitigates the errors introduced through FP4 quantization for diffusion models by adding high-precision low-rank corrections obtained via SVD. However, this high-precision low-rank correction requires many bits per rank, and moreover competes on tensor core utilization with the main FP4 GEMM. We introduce BiLoCo, a correction whose left and right factors are stored as binary $\{\pm 1\}$ sign vectors rather than as BF16 numbers. The signed format uses $16\times$ less storage per rank-one term, and since the correction now consists of additions/subtractions, this can be run on CUDA cores concurrently with the FP4 tensor-core GEMM. On 4–32B LLMs, BiLoCo matches or improves memory-matched SVDQuant for PTQ. We implement BiLoCo on B200 with a correctness-preserving CUDA schedule that runs end-to-end decode at 1.13–$1.15\times$ FP4-only, with the residual gap coming from the final output combine rather than from one-bit arithmetic.

1. Introduction

LLM decode is dominated by weight movement (Guo et al., 2024a; Lin et al., 2025; Frantar et al., 2025). Post-training quantization (PTQ) to NVFP4 on Blackwell GPUs can thus improve latency by reducing this traffic and exposing native FP4 tensor-core GEMMs. However, direct post-training quantization to FP4 still leaves an accuracy gap. A natural fix is to keep the residual weight matrix in lower bits and add a small higher-precision correction branch obtained via SVD. Such a correction approach was investigated for weight-only quantization in LQ-LoRA (Guo et al., 2024b), LoftQ (Li et al., 2023), and CALDERA (Saha et al., 2024). SVDQuant (Li et al., 2025) extends this approach to weight and activation quantization to FP4 with a BF16 rank-32 correction, where the correction absorbs weight outliers that the low-bit residual cannot represent.

The initial SVDQuant work focused on vision diffusion models. However, autoregressive LLM decode regime differs from the diffusion regime in ways that affect correction design. Concretely, with LLM decode shapes, the FP4 GEMM is weight-traffic bound: tensor cores stall on HBM while CUDA cores sit idle. A correction that uses CUDA cores rather than tensor cores could therefore be executed concurrently with the FP4 GEMM rather than competing for the same hardware. However, the high-precision low-rank correction in SVDQuant would land on tensor cores, where it would compete with the main FP4 GEMM cycles. The challenge then, is to find a correction format whose arithmetic naturally fits the CUDA-core slot.

This observation motivates Binary Low-rank Correction (BiLoCo). Instead of storing BF16 left and right singular vectors, BiLoCo stores the correction factors as $\{\pm 1\}$ sign vectors, together with scalar coefficients. Each rank-one term then reduces to signed accumulation: multiply operations are replaced by predicated add/subtract operations on CUDA cores. The representation also changes the storage trade-off. Ignoring the per-term scalar coefficient, one binary vector pair costs $16\times$ fewer bits than a BF16 vector pair, so the same vector-storage budget as a BF16 rank-k branch can store a binary rank-$16k$ branch.

The main algorithmic challenge is that the best sign-vector decomposition is not given by the SVD. We propose an efficient procedure that fits binary correction factors by greedy sign-alternation peeling and a closed-form joint least-squares solve (following Riasanovsky & Kazdadi (2024)), and then quantizes the remaining residual to NVFP4. This combination preserves the hardware-friendly sign structure while allowing the coefficients to compensate for interac-

[*]Equal contribution [1]Massachusetts Institute of Technology. Correspondence to: David Jin <jindavid@mit.edu>, Beshr IslamBouli <beshr@mit.edu>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

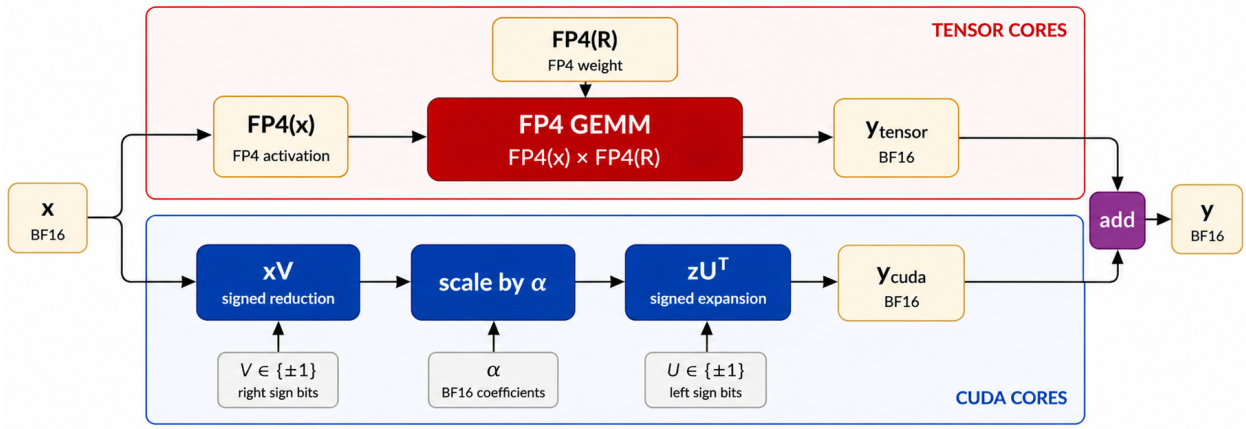

Figure 1. BiLoCo inference for performing $y = xR + xVU^\top$. A native-FP4 residual GEMM on tensor cores runs alongside a signed add/subtract correction on CUDA cores.

tions among the selected rank-one atoms.

We find that across various LLM families, BiLoCo is effective at the same storage budget as the BF16 correction used by SVDQuant. At the same vector-storage budget as a BF16 rank-32 branch, BiLoCo uses rank 512 binary factors and captures 47–53% of the Frobenius energy of LLM weight matrices across three model sizes, compared with 26–29% for BF16 rank 32. Frobenius capture is a structural proxy, so we validate model quality directly: BiLoCo matches or improves the BF16 rank-32 correction on every WikiText-2/C4 perplexity cell across four models, improves all KL mean and tail cells, and is within 0.3 pp on the WinoGrande/HellaSwag downstream protocol. On Blackwell GPUs, our custom binary-correction kernel overlaps with the native FP4 GEMM. The resulting two-branch implementation runs at only 1.13–1.15× the latency of the FP4-only kernel, with the remaining gap coming primarily from the final output combine rather than from one-bit arithmetic. Our code will be made publicly available.

2. Background

2.1. FP4 Quantization

Given the FP4 native tensor core GEMMs on the Blackwell generation of GPUs, there has been much recent work on investigating FP4 quantization. In this work we work with NVFP4 quantization, which partitions the input dimension of a tensor into blocks of 16 elements (NVIDIA, 2024). Each block is quantized to E2M1 (4-bit, 1 sign + 2 exponent + 1 mantissa) with a shared E4M3 (FP8) scale. For a weight matrix $W \in \mathbb{R}^{m\times n}$, the quantized representation is $\hat{W}_{ij} = s_{i,\lfloor j/16 \rfloor}\, q_{ij}$ with $q_{ij} \in \{0, \pm 0.5, \pm 1, \pm 1.5, \pm 2, \pm 3, \pm 4, \pm 6\}$ and s the per-block FP8 scale. The on-disk cost is approximately 4.5 bits per weight (4 weight bits plus 0.5 scale bits).

NVFP4 is attractive because both storage and arithmetic are low precision. When the activation and weight operands are in the native block-scaled FP4 format, Blackwell tensor cores can execute the main GEMM directly. However, the E2M1 codebook is extremely coarse, and each block has only one local scale. A few large entries in a block can therefore increase error for the remaining entries, while aggressive activation quantization can also amplify the effect of hidden state outliers. Thus, direct PTQ to NVFP4 can still leave a measurable accuracy gap, especially when both weights and activations are quantized.

2.2. Low-Rank Corrections for Quantization Error

One way to reduce this gap is to decompose a weight matrix into a low-bit residual plus a higher-precision correction. For weight-only quantization and efficient adaptation LQ-LoRA and LoftQ decomposes pretrained weights into a quantized component and a high-precision low-rank component via SVD, then fine-tunes the low-rank component (Guo et al., 2024b; Li et al., 2023).

SVDQuant applies this decomposition for 4-bit weight-and-activation quantization for diffusion models (Li et al., 2025). It first migrates activation outliers into the weights, then uses SVD to decompose a matrix W into a low-rank component $VU^\top$ and a residual R, which is quantized to 4 bits. Inference proceeds as

$$y = \text{FP4-GEMM}(\text{FP4}(x), \text{FP4}(R)) + xVU^\top, \quad (1)$$

This correction is effective because SVD captures directions of large residual energy that are poorly represented by the 4-bit residual. It also has a clear cost: the correction stores BF16 left and right factors and is naturally implemented as one or more higher-precision GEMMs.

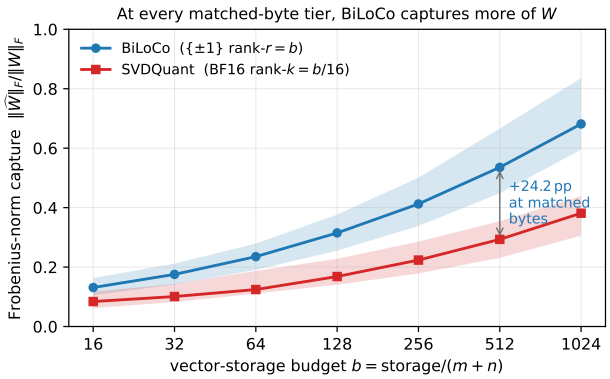

Figure 2. Frobenius-norm capture vs. vector-storage budget on Qwen3-4B projection matrices.

3. Method

3.1. Motivating Study: LLM Weights Favor Rank Multiplicity

Table 1. Equal-storage Frobenius-norm capture across three model families. Signed rank 512 and BF16 SVD rank 32 use the same vector-storage budget.

Model	σ_1/σ_{32} med.	SVD k=32 cap.	Sign r=512 cap.	Ratio	Pearson
Qwen3-4B	1.93	29.3%	52.9%	1.92×	−0.65
Qwen3-14B	2.00	26.3%	47.3%	1.77×	−0.65
Llama-3-8B	2.24	27.9%	50.6%	1.76×	−0.81

For a given memory budget, is it advantageous to have lower-rank high-precision corrections, or high-rank low-precision ones? We compare BF16 and one-bit factors at equal storage to test whether rank multiplicity beats per-atom precision. For an $m \times n$ matrix, a BF16 rank-k correction stores k left vectors and k right vectors in 16-bit precision, for a total of $16k(m+n)$ bits. A signed rank-r correction stores r sign vectors per side and a BF16 scalar vector with size r, for a total of $r(m+n+16)$ bits. This trade is useful only if the weight spectrum is flat enough that extra rank coverage beats extra per-atom precision. We test this on 189 projection matrices from Qwen3-4B, Qwen3-14B, and Llama-3-8B: 9 decoder layers per model and 7 projection types per layer. For each matrix, we compare BF16 SVD rank 32 with a signed rank-512. Table 1 shows the results, where we find that the top spectrum is flat: the median σ_1/σ_{32} ranges from 1.93 on Qwen3-4B to 2.24 on Llama-3-8B. At equal vector storage, signed rank 512 retains a median 47–53% of $\|W\|_F$, versus 26–29% for BF16 SVD rank 32. The capture ratio is anti-correlated with σ_1/σ_{32} on every family, matching the rank-multiplicity prediction.

Figure 2 plots capture against vector-storage budget b. At each matched budget, SVD uses BF16 rank k=b/16, while BiLoCo uses signed rank $r = b$. BiLoCo achieves higher capture at every tested budget. The advantage is not isolated to one projection type. On Qwen3-4B, the matched-byte signed/SVD capture ratio ranges from 1.4× to 2.0× across q/k/v/o/gate/up/down projections, with the full breakdown in Table 6. This motivates BiLoCo: the same correction bytes can cover many more directions if the factors are one-bit signs. Since Frobenius capture is only a structural proxy, Section 4 evaluates whether this storage advantage preserves model quality.

3.2. BiLoCo: Binary Low-Rank Corrections for Quantized Weights

BiLoCo uses the same two-branch inference form as SVDQuant (Eq. 1), but changes the representation of the correction branch. Instead of a BF16 factorization, it stores signed rank-one terms and one scalar coefficient per term. At PTQ time, for each weight matrix W, we choose a correction $C = \sum_i \alpha_i u_i v_i^\top$ and store the residual $R = W - C$ in NVFP4. NVFP4 uses 16-element input-channel blocks, E2M1 4-bit values, and a shared FP8 scale, for approximately 4.5 bits per residual weight. We represent the correction as

$$W \approx \sum_{i=1}^{r} \alpha_i\, u_i v_i^\top + R, \qquad u_i \in \{\pm 1\}^m, \quad v_i \in \{\pm 1\}^n, \quad \alpha_i \in \mathbb{R}. \tag{2}$$

The deployed forward pass, with BF16 coefficient α, is

$$y = \text{FP4-GEMM}(\text{FP4}(x), \text{FP4}(R)) + \sum_{i=1}^{r} \alpha_i\, u_i\, (v_i^\top x), \tag{3}$$

Importantly, each $v_i^\top x$ is a sum-of-conditional-sign products and contains no multiplications, i.e., $\sum_k v_{i,k}\, x_k$ where $v_{i,k} \in \{\pm 1\}$ selects between $+x_k$ and $-x_k$. This extends to the out-product with u_i. This maps naturally to CUDA-core add/subtract pipelines and is complementary to the tensor-core FP4 GEMM.

To find the binary low-rank corrections, we avoid an explicit SVD and incorporate a modified signed-cut decomposition algorithm (Frieze & Kannan, 1999; Riasanovsky & Kazdadi, 2024) to build $\{u_i, v_i, \alpha_i\}_{i=1}^r$ greedily. We first initialize $R \leftarrow W$. Then for each rank-1 component, we draw a random sign vector v, then alternate

$$u \leftarrow \text{sign}(Rv), \qquad v \leftarrow \text{sign}(R^\top u) \tag{4}$$

for T iterations. The optimal scalar for fixed binary u, v is $\alpha = (u^\top R v)/(mn)$, which minimizes $\|R - \alpha u v^\top\|_F^2$ subject to u, v being binary. We then deflate via $R \leftarrow R - \alpha u v^\top$ and proceed on to the next rank. In practice, $T = 5$ is sufficient for the alternation to settle.

Algorithm 1 Greedy binary peeling with global coefficient refit

Require: Weight matrix $W \in \mathbb{R}^{m\times n}$, rank r, alternation steps T
$R \leftarrow W$
for $i = 1, \ldots, r$ **do**
  initialize $v_i \in \{\pm 1\}^n$ at random
  for $t = 1, \ldots, T$ **do**
    $u_i \leftarrow \mathrm{sign}(Rv_i)$
    $v_i \leftarrow \mathrm{sign}(R^\top u_i)$
  end for
  $\alpha_i \leftarrow (u_i^\top R v_i)/(mn)$
  $R \leftarrow R - \alpha_i u_i v_i^\top$
end for
Form $A_{ij} = (u_i^\top u_j)(v_i^\top v_j)$ and $b_i = u_i^\top W v_i$
Solve $(A + \epsilon I)\alpha = b$ and set $R \leftarrow \mathrm{FP4}(W - \sum_i \alpha_i u_i v_i^\top)$
return sign atoms $\{u_i, v_i\}_{i=1}^r$, BF16 coefficients α, and FP4 residual R

Greedy peeling produces (u_i, v_i) at a coordinate-wise stationary point per term but uses the per-term closed form $\alpha_i = (u_i^\top R_i v_i)/(mn)$ that ignores cross-term correlations. Once all r signed terms are produced, we refit $\boldsymbol{\alpha} = \{\alpha_i\}_{i=1}^r$ jointly in closed form:

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^r} \left\| W - \textstyle\sum_i \alpha_i u_i v_i^\top \right\|_F^2, \tag{5}$$

which solves the $r \times r$ system $A\boldsymbol{\alpha} = b$ with $A_{ij} = (u_i^\top u_j)(v_i^\top v_j)$ and $b_i = u_i^\top W v_i$. A small regularization ϵI (i.e., as in ridge regression) is added for numerical stability. The solve is $O(r^3)$ and runs in microseconds per layer; building b shares the $U^\top W V$ matmul that the peel step already amortizes. After the refit, the storage residual $R_{\text{store}} = W - U\mathrm{diag}(\alpha)V^\top$ is quantized to NVFP4 once. Algorithm 1 summarizes the full peeling and refit procedure.

3.3. Theoretical Analysis

BiLoCo uses greedy sign-alternation peeling because each coordinate update maximizes the current bilinear objective over one set of signs, and therefore monotonically improves the single-term Frobenius decrease. After the signs are fixed, the final coefficient refit is the least-squares optimum over the selected atoms. Our theoretical analysis shows that the PTQ solver is a principled matching-pursuit procedure for the Frobenius norm as a proxy objective.

Theorem 3.1 (Monotone single-term sign alternation) *Let $R \in \mathbb{R}^{m\times n}$ be fixed. Starting from any $v^{(0)} \in \{\pm 1\}^n$, alternate $u^{(t)} = \mathrm{sign}(Rv^{(t-1)})$ and $v^{(t)} = \mathrm{sign}(R^\top u^{(t)})$, with deterministic tie-breaking. Then $u^\top R v$ is non-decreasing after each coordinate update. For the final signs u, v, the optimal scalar is $\alpha^\star = (u^\top R v)/(mn)$, and deflation yields*

$$\|R\|_F^2 - \|R - \alpha^\star u v^\top\|_F^2 \;=\; \frac{(u^\top R v)^2}{mn}.$$

Proposition 3.2 (Global coefficient refit) *For fixed sign matrices $U \in \{\pm 1\}^{m\times r}$ and $V \in \{\pm 1\}^{n\times r}$, the least-squares coefficients*

$$\alpha^\star = \arg\min_\alpha \left\| W - \sum_i \alpha_i u_i v_i^\top \right\|_F^2$$

solve $A\alpha = b$, where

$$A_{ij} = (u_i^\top u_j)(v_i^\top v_j), \qquad b_i = u_i^\top W v_i.$$

Replacing the greedy coefficients by $\alpha^\star$ cannot increase the Frobenius residual.

Optional cut-norm motivation. The sign atoms can also be viewed through a signed-cut lens. Let $\|R\|_\blacksquare$ denote the signed cut norm. If, at each peeling step t, the selected signs satisfy

$$|u_t^\top R_t v_t| \geq \rho \, \|R_t\|_\blacksquare$$

for some $\rho \in (0, 1]$, then with $\alpha_t = (u_t^\top R_t v_t)/(mn)$ and $R_{t+1} = R_t - \alpha_t u_t v_t^\top$, the residuals satisfy

$$\min_{0\leq t<r} \|R_t\|_\blacksquare^2 \;\leq\; \frac{mn}{\rho^2 r} \|W\|_F^2.$$

This is a conditional Frieze–Kannan-style statement (Frieze & Kannan, 1999; Alon & Naor, 2004): it explains what a uniformly good signed-cut oracle would imply, but it is not a claim that our alternating heuristic attains a fixed approximation ratio to the optimal cut norm. Empirically, we measure the Frobenius-relative capture rate

$$\rho_F(t) = \frac{|u_t^\top R_t v_t|}{\sqrt{mn}\, \|R_t\|_F},$$

which is exactly the rate constant in

$$\|R_{t+1}\|_F^2 = \|R_t\|_F^2 \left(1 - \rho_F(t)^2\right).$$

Across 63 Qwen3-4B matrices and 8064 measured atom updates, ρ_F remains bounded away from zero in the diagnostic sweep. The full ρ_F distribution is reported in Appendix A.2.

What the analysis is and is not. The analysis establishes three local facts: the per-term sign search is monotone, the deployed coefficients are optimal over the chosen signs, and a conditional cut-norm oracle would yield a standard residual-decay statement. It does not bound output error, downstream accuracy, or PPL. The role of the analysis is to support the storage–accuracy argument from Section 3.1 and to make the greedy signed-peeling solver principled rather than ad hoc; the model-quality claims are validated empirically in Section 4.

3.4. Kernel Design

We develop a custom kernel to realize the efficiency benefits of BiLoCo. The kernel targets LLM decode, where the residual FP4 GEMM is dominated by weight movement. Our measurements use a representative B200 shape with $M{=}1$ and $K{=}N{=}5120$; larger decode batches are discussed in Section 6. For a layer with $W \in \mathbb{R}^{N\times K}$, input $X \in \mathbb{R}^{M\times K}$, and output $Y \in \mathbb{R}^{M\times N}$,

$$Y = \underbrace{X R^\top}_{\text{tensor-core path}} + \underbrace{\sum_{i=1}^{r} \alpha_i (X v_i) u_i^\top}_{\text{BiLoCo branch}}, \qquad u_i \in \{\pm 1\}^N, \quad v_i \in \{\pm 1\}^K. \tag{6}$$

Runtime representation and memory scale. Both sign factors remain bit-packed at runtime: $V \in \{0,1\}^{r\times K/8}$ and $U \in \{0,1\}^{r\times N/8}$, with $1 \leftrightarrow +1$ and $0 \leftrightarrow -1$. Rows of V are contiguous for the input projection; packed U is read by output tiles in the expansion kernel. The intermediate rank vector and the persistent output accumulator are FP32, and the branch delta is cast to BF16 before the output combine. At $r{=}512$, the branch reads roughly $r(K+N)/8 + 2K \approx 0.65$ MiB for signs and the BF16 token, while the FP4 residual path reads about 12.5 MiB of E2M1 weights plus 1.6 MiB of scales. The branch therefore moves over $20\times$ less data than the FP4 path; the measured microsecond-scale time comes from reductions, atomics, and graph/kernel overhead rather than DRAM bandwidth.

Branch kernels. The branch has two kernels. First, xV computes $z_i = \alpha_i (v_i^\top x)$ for each rank

$$v_i^\top x = 2 \sum_{k:V_{i,k}=1} x_k - \sum_{k=1}^{K} x_k, \tag{7}$$

so $\sum_k x_k$ is computed once and reused across ranks; the inner loop then conditionally includes only the positive bits, roughly halving the per-rank additions. Second, zU expands z through packed U using the same identity over the rank dimension. We tile the output dimension and split rank into groups, which exposes many independent CTAs even at $M{=}1$. In the development sweep, changing the rank-group size from 8 to 4 doubled CTA count for the $r{=}32$ decode case and reduced branch latency by about 25%. Each CTA accumulates its tile and atomic-adds into an FP32 accumulator $A \in \mathbb{R}^N$.

Persistent accumulator and scheduling. To avoid clearing A on every token, the decode kernel keeps it persistent and emits the token-local delta $A^{(t)} - Y_{\text{prev}}$; both buffers are reset when a sequence ends or a decode slot is reused. This costs $8N$ bytes per layer per active slot, about 40 KiB for $N{=}5120$. The FP4 residual GEMM and the BiLoCo branch run on separate CUDA streams; after both streams finish, a small BF16 kernel combines their outputs. The *join/no-add* timing in Section 4.2 tests whether the branch hides before this combine, while the *full* timing includes the correctness-preserving combine. We exclude a race-residual variant that was bit-exact in trials but lacked a correct kernel-level ordering point. The remaining fusion target is to fold the combine into the FP4 mainloop or epilogue.

4. Experiments

We test the two halves of the §1 hypothesis in turn. §4.1 asks whether the matched-storage correction preserves model quality, using perplexity, held-out KL, and downstream tasks. §4.2 then asks whether the one-bit branch fits inside the native FP4 decode envelope, using the correctness-preserving two-stream schedule from §3.4. Kernel component attribution, discarded synchronization variants, per-task downstream results, and KL at additional tiers are deferred to Appendix A.

4.1. Performance Evaluation

Setup. We evaluate on Qwen3-4B, Llama-3-8B, Qwen3-14B and Qwen3-32B. All quantization uses SmoothQuant (Xiao et al., 2024) presmoothing with $\alpha = 0.5$, $T = 5$ greedy peeling iterations, and the "four over six" NVFP4 per-block MSE scale search following Cook et al. (2025) for both weights and activations. For $r = 512$, the full BiLoCo PTQ pass takes approximately 1 minute for Qwen3-4B and 5 minutes for Qwen3-32B on a single B200/B300. Calibration uses 128 sequences from C4 train dataset at length 2048. WikiText-2 test is concatenated and split into non-overlapping 2048-token windows, and C4 validation uses 256 fixed-seed random 2048-token windows. Compute resources are summarized in Appendix A.1.

Three accuracy measurements at the matched-storage tier $b{=}512$ across four model sizes: perplexity (Table 2), held-out KL divergence to the BF16 reference (Table 3), and downstream zero-shot mean accuracy (Table 4). We also report BiLoCo at $r = 128$, whose Frobenius capture is comparable to SVDQuant $k = 32$ in Figure 2. The kernel rank sweep, the per-task downstream breakdown, and KL at non-headline tiers are in Appendix A.

Perplexity. For all four models, BiLoCo entries average results from six sign initializations. SVDQuant is deterministic for the fixed calibration batch. At $b{=}512$, BiLoCo (r=512) has lower PPL than SVDQuant (k=32) on the six Qwen3-4B/14B and Llama-3-8B WikiText-2/C4 cells. The three largest margins (Qwen3-14B C4 and both Llama-3-8B datasets) resolve beyond 2 standard errors of the BiLoCo mean; the Qwen3-4B and Qwen3-14B WikiText-2 cells are

Table 2. Matched-storage perplexity at b=512 (lower is better). BiLoCo entries report mean ± std over sign initializations. Bold marks the best corrected FP4 method, with ties bolded.

Model	Dataset	BF16	NVFP4	SVDQuant k=32	BiLoCo r=128	BiLoCo r=512
Qwen3-4B	Wiki2	14.22	15.46	15.07	15.09 ±0.58	**14.94** ±0.37
Qwen3-4B	C4	15.25	16.05	15.85	15.88 ±0.23	**15.81** ±0.17
Llama-3-8B	Wiki2	6.27	6.98	6.92	6.93 ±0.01	**6.84** ±0.01
Llama-3-8B	C4	6.69	7.70	7.59	7.61 ±0.01	**7.48** ±0.01
Qwen3-14B	Wiki2	8.89	9.41	9.19	9.21 ±0.03	**9.17** ±0.04
Qwen3-14B	C4	11.19	11.58	11.46	11.49 ±0.03	**11.44** ±0.02
Qwen3-32B	Wiki2	7.85	8.21	8.18	8.22 ±0.03	**8.16** ±0.03
Qwen3-32B	C4	10.09	10.46	**10.40**	10.43 ±0.01	**10.40** ±0.02

directionally positive but within the multi-seed noise floor. On Qwen3-32B, BiLoCo is slightly better than SVDQuant on WikiText-2 and tied on C4 within the seed noise. At r=128, BiLoCo is roughly on par with SVDQuant while using one quarter of the vector-storage budget.

Held-out KL. We compare the quantized next-token distributions to the BF16 reference on 200 held-out Fineweb samples (Table 3). At b=512, BiLoCo r=512 wins all eight KL-mean/KL-p99 cells across the four models. KL is a stricter test than PPL because it penalizes both center and tail miscalibration of the predictive distribution; that BiLoCo wins every cell—and that tail margins (e.g. Qwen3-14B p99 0.130 vs. 0.167, -22% relative) exceed mean margins—supports that the b=512 accuracy advantage is not a metric artifact. KL at the F-norm-capture-equivalent tier b=128 is reported in Appendix A.7.

Downstream tasks. Table 4 reports the WinoGrande (5-shot) and HellaSwag (10-shot) mean accuracy used by the recent NVFP4 study (Egiazarian et al., 2026). At b=512, BiLoCo improves SVDQuant on three of four models (Qwen3-4B, Qwen3-14B, and Qwen3-32B by 0.1–0.4 pp) and trails Llama-3-8B by 0.3 pp. At the F-norm-capture-equivalent tier b=128, BiLoCo r=128 stays within 0.3 pp of SVDQuant k=32 on every model while using one quarter of the vector-storage budget. The standard five-task PTQ battery (ARC-easy, ARC-challenge, PIQA, WinoGrande, OBQA) is reported per-task in Appendix A.6; there BiLoCo r=512 wins the mean on Qwen3-4B and Qwen3-14B and trails Llama-3-8B by 0.003, while r=128 wins the Llama-3-8B mean.

4.2. Decode Kernel Efficiency

Setup. Kernel measurements run on NVIDIA B200 GPUs (compute capability 10.0) at the decode shape M=1, K=N=5120. The residual path uses native NVFP4, and all timed schedules are captured as CUDA Graphs to amortize Python dispatch overhead. The implementation follows §3.4: native FP4 on one stream, the one-bit BiLoCo branch on a second stream, and a BF16 output combine after both streams join.

Table 5 is the main systems result. For BiLoCo, the branch finishes before the FP4-only wall-clock on every target rank, including the matched-storage r=512 row (12.93 μs branch vs. 15.65 μs FP4). The diagnostic join/no-add timing is therefore essentially the FP4 baseline: both streams finish by 15.5–15.7 μs. The correctness-preserving full schedule is still 1.13–1.15× FP4-only because the final BF16 combine remains unfused. Across ranks, this post-add is 1.95–2.11 μs and accounts for almost all of $T_{\text{full}} - T_{\text{FP4}}$. The small branch-time jump between r=64 and r=128 comes from the rank-grouped zU expansion crossing into additional rank-group work; it remains below the FP4 stream. The corresponding fusion target is to fold the final combine into the FP4 epilogue or mainloop while keeping the rank-parallel one-bit branch.

The SVDQuant rows store dense BF16 low-rank factors and compute the branch with two BF16 GEMMs. Their branch also hides under FP4 at decode-5k, but the full two-stream reference is about 1.25–1.27× FP4 because it pays the same post-join output combine. Appendix Table 10 gives the full dense-BF16 sweep with the more pessimistic CUDA-core fused reference.

Correctness was checked against an unfused one-bit reference for ranks 16–512 over repeated CUDA-Graph replays. All rows had zero elements above BF16-space tolerance 0.25, with maximum observed error at most 0.125 in the packed-U replay checks. Appendix A.4 reports component attribution and failed synchronization variants, including cooperative-grid and split seed/add/finalize schedules whose ordering was correct but whose overhead erased the no-clear benefit.

Table 3. Held-out KL to the BF16 reference at b=512 on Fineweb samples. Lower is better.

	KL mean (↓)		KL p99 (↓)	
Model	BiLoCo r=512	SVDQuant k=32	BiLoCo r=512	SVDQuant k=32
Qwen3-4B	**0.0740**	0.0835	**0.1421**	0.1649
Qwen3-14B	**0.0480**	0.0517	**0.1303**	0.1671
Llama-3-8B	**0.0756**	0.0824	**0.3252**	0.3409
Qwen3-32B	**0.0590**	0.0605	**0.3175**	0.3363

Table 4. Downstream mean accuracy on the WinoGrande/HellaSwag protocol. Higher is better.

Method	Qwen3-4B	Qwen3-14B	Llama-3-8B	Qwen3-32B
BF16 (reference)	0.686	0.772	0.803	0.803
SVDQuant k=32 (b=512)	0.669	0.759	**0.780**	0.796
BiLoCo r=128 (b=128)	0.667	**0.763**	0.777	0.795
BiLoCo r=512 (b=512)	**0.670**	**0.763**	0.777	**0.799**

5. Related work

Post-training quantization for LLMs. GPTQ (Frantar et al., 2023) and AWQ (Lin et al., 2026) target W4A16 (4-bit weights, bf16 activations); SmoothQuant (Xiao et al., 2024) introduced the per-channel scale migration we use as a preprocessing step. QuaRot (Ashkboos et al., 2024) and SpinQuant (Liu et al., 2025) are rotation-based W4A4 methods that pre-rotate weights and activations to make them more quantization-friendly. Rotation is orthogonal to correction format: our binary branch can be applied on top of a QuaRot backbone, and we leave the empirical evaluation of this composition to future work.

Low-rank corrections for FP4 quantization. SVDQuant (Li et al., 2025) introduced the FP4-with-low-rank-correction recipe for diffusion models, with a BF16 rank-32 projector as the default correction. We extend this pattern to LLMs and study an alternative format under the same byte budget: at the storage cost of a BF16 rank-32 projector, $16 \times 32 = 512$ ± 1 atoms can be fit instead. The structural argument for this swap (Section 3.1) is that LLM weights have flat top spectra, so rank multiplicity wins over per-atom precision at fixed bytes. LQ-LoRA (Guo et al., 2024b) explored low-rank + low-bit decomposition for fine-tuning, focused on training-time storage rather than inference wall-clock; the byte-budget framing applies there too but with different deployment constraints.

1-bit and ternary networks. BitNet (Wang et al., 2023) and 1.58-bit LLMs (Zhang et al., 2026) train models entirely in $\{\pm 1\}$ or $\{\pm 1, 0\}$ weights from scratch; they are not post-training corrections to a higher-precision model. We borrow the binary representation but apply it as an additive PTQ branch on top of FP4.

Signed cut decompositions on LLM weights. The closest line of work (Riasanovsky & Kazdadi, 2024) applies Frieze–Kannan-style signed cut decompositions to LLM weight matrices. Their Algorithm 5 (greedy sign-alternation peel) is the same per-atom search used inside our Algorithm 1, and their Algorithm 6 is the global α refit. Our contribution is positioned differently: we use sign atoms as a small-rank *additive* correction to an FP4 main path, framed by the matched-storage argument of Section 3.1, and implement the correction as a Blackwell CUDA correction designed to overlap with native FP4. Riasanovsky & Kazdadi (2024) use sign atoms at width $w \sim 10^4$ as a 50% spatial replacement for BF16 (CPU-side, Rust + AVX-512); ours is a Blackwell CUDA-core branch paired with an FP4 tensor-core main path.

Spectral structure of LLM weights. Prior work has used singular-value spectra to analyze or compress transformer weights: random-matrix analyses study where information is stored across the spectrum, and SVD-based compression methods quantify the loss from truncating singular components (Staats et al., 2025; Wang et al., 2025b;a; Yuan et al., 2025). Our Section 3.1 measurement is complementary: rather than using SVD for compression directly, we measure the top-spectrum shape at the SVDQuant rank and find that it is flat enough that a fixed correction budget is better spent on many signed atoms than on fewer high-precision atoms.

Heterogeneous execution on GPUs. Recent GPU kernels increasingly treat hardware utilization as a first-class design objective, overlapping matmul, memory movement, and non-matmul work in attention and serving systems (Shah et al., 2024; Zadouri et al., 2026; Lin et al., 2025; Zhao et al., 2023). BiLoCo applies the same principle at the layer level: the residual GEMM remains on FP4 tensor cores, while the binary correction runs as signed reductions and expansions on CUDA cores.

Table 5. Decode-kernel comparison on B200 at decode-5k, aligned by storage budget b. For BiLoCo, $b{=}r$ and both sign factors remain one-bit at runtime. For SVDQuant, $b{=}16k$ and the branch is a dense-BF16 two-GEMM reference, not the official SVDQuant/Nunchaku kernel. *Join/no-add* waits for both streams but omits the output combine. Times are CUDA-Graph medians in μs.

b	FP4 only	BiLoCo ($r{=}b$)				SVDQuant ($k{=}b/16$)				
		branch	join/no-add	full	full/FP4	k	branch	join/no-add	full	full/FP4
16	15.55	9.38	15.65	17.66	1.136×	1	9.02	17.54	19.62	1.262×
32	15.39	9.57	15.65	17.70	1.150×	2	10.50	17.54	19.58	1.272×
64	15.65	9.47	15.55	17.66	1.129×	4	10.62	17.50	19.58	1.251×
128	15.62	11.23	15.55	17.66	1.131×	8	11.20	17.12	19.58	1.254×
256	15.58	11.33	15.65	17.70	1.136×	16	11.62	17.50	19.62	1.259×
512	15.65	12.93	15.71	17.66	1.129×	32	12.08	17.50	19.55	1.249×

6. Limitations and future work

Production decode kernels. Our B200 kernel validates the heterogeneous design at the decode shape $M{=}1, K{=}N{=}5120$, but it is not yet a full production serving kernel. The next systems step is a fused decode implementation across larger small-batch regimes, with the final output combine folded into the FP4 mainloop or epilogue and with explicit per-slot accumulator management.

Composition with rotations. All experiments use SmoothQuant presmoothing and C4-train calibration. We have not tested BiLoCo on rotation-based W4A4 backbones such as QuaRot or SpinQuant, nor have we swept activation-aware refinement or out-of-distribution calibration. These are natural extensions because the binary correction format is orthogonal to the choice of quantization backbone.

Broader Impact. BiLoCo can reduce the memory and latency cost of LLM inference, which may lower serving costs and energy use. As with other LLM efficiency methods, reducing inference cost also lowers the marginal cost of deploying existing models, including for misuse. BiLoCo does not introduce new model capabilities or release a new pretrained model.

7. Conclusion

We presented BiLoCo, a one-bit additive correction for FP4 LLM decode that replaces a few BF16 rank-one terms with many signed atoms at the same vector-storage budget. This representation matches the flat-spectrum structure of LLM weights and maps the correction to CUDA cores while the FP4 residual GEMM runs on tensor cores. The broader lesson is that low precision per atom can buy both rank multiplicity and a better hardware schedule.

References

Alon, N. and Naor, A. Approximating the cut-norm via grothendieck's inequality. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '04, pp. 72–80, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138520. doi: 10.1145/1007352.1007371. URL https://doi.org/10.1145/1007352.1007371.

Ashkboos, S., Mohtashami, A., Croci, M. L., Li, B., Cameron, P., Jaggi, M., Alistarh, D., Hoefler, T., and Hensman, J. Quarot: Outlier-free 4-bit inference in rotated llms, 2024. URL https://arxiv.org/abs/2404.00456.

Cook, J., Guo, J., Xiao, G., Lin, Y., and Han, S. Four over six: More accurate nvfp4 quantization with adaptive block scaling, 2025. URL https://arxiv.org/abs/2512.02010.

Egiazarian, V., Castro, R. L., Kuznedelev, D., Panferov, A., Kurtic, E., Pandit, S., Marques, A., Kurtz, M., Ashkboos, S., Hoefler, T., and Alistarh, D. Bridging the gap between promise and performance for microscaling fp4 quantization, 2026. URL https://arxiv.org/abs/2509.23202.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. Gptq: Accurate post-training quantization for generative pre-trained transformers, 2023. URL https://arxiv.org/abs/2210.17323.

Frantar, E., Castro, R. L., Chen, J., Hoefler, T., and Alistarh, D. Marlin: Mixed-precision auto-regressive parallel inference on large language models. In *Proceedings of the 30th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, PPoPP '25, pp. 239–251, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400714436. doi: 10.1145/3710848.3710871. URL https://doi.org/10.1145/3710848.3710871.

Frieze, A. and Kannan, R. Quick approximation to matrices and applications. *Combinatorica*, 19(2):175–220, Feb 1999. ISSN 1439-6912. doi: 10.1007/s004930050052. URL https://doi.org/10.1007/s004930050052.

Guo, H., Brandon, W., Cholakov, R., Ragan-Kelley, J., Xing, E. P., and Kim, Y. Fast matrix multiplications for lookup table-quantized LLMs. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 12419–12433, Miami, Florida, USA, November 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.724. URL https://aclanthology.org/2024.findings-emnlp.724/.

Guo, H., Greengard, P., Xing, E. P., and Kim, Y. Lq-lora: Low-rank plus quantized matrix decomposition for efficient language model finetuning, 2024b. URL https://arxiv.org/abs/2311.12023.

Li, M., Lin, Y., Zhang, Z., Cai, T., Guo, J., Li, X., Xie, E., Meng, C., Zhu, J.-Y., and Han, S. SVDQuant: Absorbing outliers by low-rank component for 4-bit diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=vWR3KuiQur.

Li, Y., Yu, Y., Liang, C., He, P., Karampatziakis, N., Chen, W., and Zhao, T. Loftq: Lora-fine-tuning-aware quantization for large language models. *arXiv preprint arXiv:2310.08659*, 2023.

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.-M., Wang, W.-C., Xiao, G., Dang, X., Gan, C., and Han, S. Awq: Activation-aware weight quantization for llm compression and acceleration, 2026. URL https://arxiv.org/abs/2306.00978.

Lin, Y., Tang, H., Yang, S., Zhang, Z., Xiao, G., Gan, C., and Han, S. QServe:w4a8KV4 quantization and system co-design for efficient LLM serving. In *Eighth Conference on Machine Learning and Systems*, 2025. URL https://openreview.net/forum?id=1FfmStySS1.

Liu, Z., Zhao, C., Fedorov, I., Soran, B., Choudhary, D., Krishnamoorthi, R., Chandra, V., Tian, Y., and Blankevoort, T. Spinquant: Llm quantization with learned rotations, 2025. URL https://arxiv.org/abs/2405.16406.

NVIDIA. Introducing NVFP4 for Efficient and Accurate Low-Precision Inference | NVIDIA Technical Blog — developer.nvidia.com. https://developer.nvidia.com/blog/introducing-nvfp4-for-efficient-and-accurate-low-precision-inference, 2024. [Accessed 05-05-2026].

Riasanovsky, A. W. N. and Kazdadi, S. E. Efficient 1-bit tensor approximations, 2024. URL https://arxiv.org/abs/2410.01799.

Saha, R., Sagan, N., Srivastava, V., Goldsmith, A. J., and Pilanci, M. Compressing large language models using low rank and low precision decomposition, 2024. URL https://arxiv.org/abs/2405.18886.

Shah, J., Bikshandi, G., Zhang, Y., Thakkar, V., Ramani, P., and Dao, T. Flashattention-3: Fast and accurate attention with asynchrony and low-precision, 2024. URL https://arxiv.org/abs/2407.08608.

Staats, M., Thamm, M., and Rosenow, B. Small singular values matter: A random matrix analysis of transformer models, 2025. URL https://arxiv.org/abs/2410.17770.

Wang, H., Ma, S., Dong, L., Huang, S., Wang, H., Ma, L., Yang, F., Wang, R., Wu, Y., and Wei, F. Bitnet: Scaling 1-bit transformers for large language models, 2023. URL https://arxiv.org/abs/2310.11453.

Wang, X., Alam, S., Wan, Z., Shen, H., and Zhang, M. Svd-llm v2: Optimizing singular value truncation for large language model compression, 2025a. URL https://arxiv.org/abs/2503.12340.

Wang, X., Zheng, Y., Wan, Z., and Zhang, M. Svd-llm: Truncation-aware singular value decomposition for large language model compression, 2025b. URL https://arxiv.org/abs/2403.07378.

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., and Han, S. Smoothquant: Accurate and efficient post-training quantization for large language models, 2024. URL https://arxiv.org/abs/2211.10438.

Yuan, Z., Shang, Y., Song, Y., Yang, D., Wu, Q., Yan, Y., and Sun, G. Asvd: Activation-aware singular value decomposition for compressing large language models, 2025. URL https://arxiv.org/abs/2312.05821.

Zadouri, T., Hoehnerbach, M., Shah, J., Liu, T., Thakkar, V., and Dao, T. Flashattention-4: Algorithm and kernel pipelining co-design for asymmetric hardware scaling, 2026. URL https://arxiv.org/abs/2603.05451.

Zhang, D., Wu, X., Huang, S., Wang, Y., Shao, H., Hao, Y., Chi, Z., Dong, L., Song, T., Xia, Y., Sui, Z., and Wei, F. Sparse-bitnet: 1.58-bit llms are naturally friendly to semi-structured sparsity, 2026. URL https://arxiv.org/abs/2603.05168.

Zhao, H., Cui, W., Chen, Q., and Guo, M. Ispa: Exploiting intra-sm parallelism in gpus via fine-grained resource management. *IEEE Transactions on Computers*, 72(5):1473–1487, 2023. doi: 10.1109/TC.2022.3214088.

Table 6. Per-projection Frobenius-norm capture $\mathrm{capA}(r) = \|\widehat{W}_r\|_F / \|W\|_F$ at three operating points: SVDQuant rank r=32 (b=512), and BiLoCo ranks r=128 (b=128, quarter storage of SVDQuant r=32) and r=256 (b=256, half storage). Median across 9 evenly-spaced decoder layers per model. Bold marks BiLoCo r=256 (it wins every cell on every model). Higher is better.

Model	Projection	SVDQuant r=32	BiLoCo r=128	BiLoCo r=256
Qwen3-4B	q_proj	28.1%	35.4%	**44.6%**
	k_proj	36.9%	43.5%	**55.8%**
	v_proj	24.7%	38.3%	**51.8%**
	o_proj	21.5%	30.8%	**41.7%**
	gate_proj	18.5%	26.5%	**34.9%**
	up_proj	15.0%	23.4%	**32.3%**
	down_proj	17.4%	24.1%	**33.0%**
Qwen3-14B	q_proj	32.1%	31.0%	**39.5%**
	k_proj	39.7%	40.9%	**53.1%**
	v_proj	29.0%	35.1%	**47.9%**
	o_proj	25.0%	25.5%	**34.8%**
	gate_proj	22.6%	22.0%	**28.6%**
	up_proj	16.5%	17.4%	**24.0%**
	down_proj	18.1%	19.3%	**26.0%**
Llama-3-8B	q_proj	35.1%	34.4%	**44.0%**
	k_proj	41.3%	43.2%	**55.5%**
	v_proj	30.7%	35.6%	**48.5%**
	o_proj	27.3%	28.3%	**38.5%**
	gate_proj	24.6%	24.5%	**31.1%**
	up_proj	17.4%	19.4%	**26.6%**
	down_proj	19.3%	20.5%	**27.6%**

A. Additional Experimental Details

A.1. Compute resources

Reported experiments use single-GPU Modal B200/B300-class workers. Accuracy and PTQ runs allocate 8–16 CPU cores and 96–256 GiB host memory, increasing to 128–256 GiB for Qwen3-32B BiLoCo runs. Downstream sweeps are capped at two concurrent workers. Kernel microbenchmarks use the same GPU class with 64–256 GiB host memory and one-hour job limits. The largest observed accuracy run peaks at 136 GiB CUDA memory; the reported r=512 PTQ pass takes approximately one minute on Qwen3-4B and five minutes on Qwen3-32B. Additional exploratory kernel attempts were run during development and are not included in the reported experimental totals.

A.2. Structural and cut-premise diagnostics

Aggregating the per-matrix capture diagnostic of Section 3.1 by projection type confirms that the binary advantage appears in every projection of every layer, on every model, rather than in one outlier module. Numbers in Table 6 are medians across 9 evenly-spaced decoder layers per model; the metric is the metric-A Frobenius capture $\mathrm{capA}(r) = \|\widehat{W}_r\|_F / \|W\|_F$ from Table 1, where $\widehat{W}_r$ is the rank-r truncated SVD reconstruction (SVDQuant column) or the rank-r greedy signed-peel reconstruction (BiLoCo columns).

Two takeaways. (i) *BiLoCo r=256 dominates SVDQuant r=32 on every cell of every model* (21/21 projections), while using only half the vector-storage budget. (ii) At b=128 (quarter storage), BiLoCo captures roughly the same Frobenius energy as SVDQuant on Qwen3-14B and Llama-3-8B, and substantially more on Qwen3-4B; the differing spectral flatness across families (Table 1) explains the per-model gap pattern.

The second diagnostic is the empirical premise plot for the conditional cut-decay theorem. Theorem B.2's premise involves the cut norm $\|R_t\|_{\blacksquare}$, which is NP-hard to compute. We instead measure the Frobenius-relative analog $\rho_F(t) = |u_t^\top R_t v_t| / (\sqrt{mn}\,\|R_t\|_F)$, which is exactly the rate constant in $\|R_{t+1}\|_F^2 = \|R_t\|_F^2 (1 - \rho_F(t)^2)$ and lower-bounds the cut-norm condition by Cauchy–Schwarz. Figure 3 reports the per-iteration distribution across 63 Qwen3-4B matrices: $\rho_F \geq 0.019$ uniformly across all 8064 atom updates, and the median decays only mildly from 0.036 at iteration 0 to 0.026 at iteration 127. The premise activates with a non-trivial floor in practice. Boxes show interquartile ranges, whiskers show q5/q95, and iterations are binned in groups of 8 for readability.

A.3. Rank sweep

Table 7 checks rank monotonicity on Qwen3-4B and Qwen3-14B using three sign initializations per rank.

Reading guide. The net trend from r=8 to r=1024 decreases on every column. There are eight small non-monotonic ticks across 32 rank-to-rank steps, all within multi-seed noise. Qwen3-14B is the cleaner sweep: its standard deviation is at most 0.08, compared with up

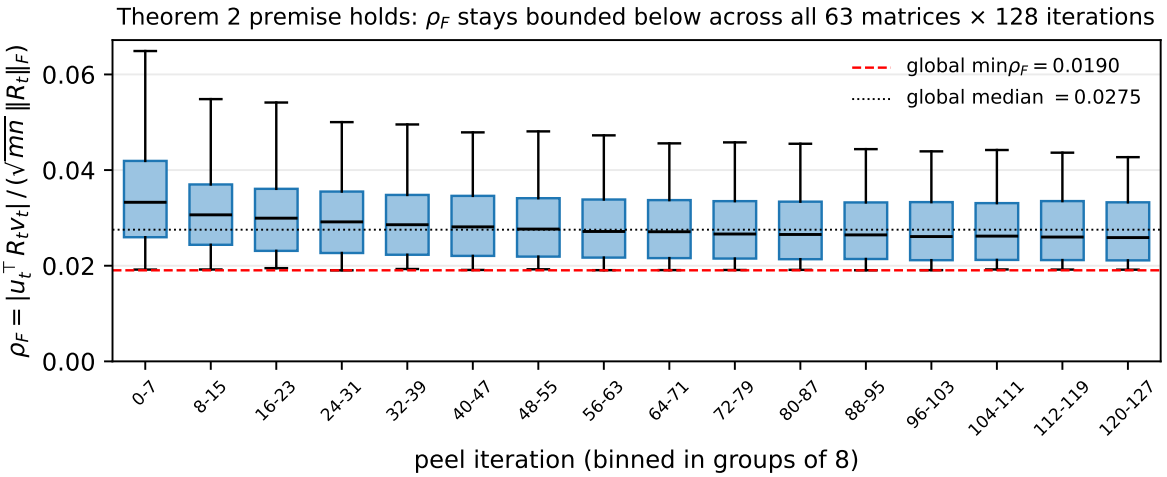

Figure 3. Frobenius-relative capture rate over 128 peel iterations on Qwen3-4B projection matrices.

Table 7. BiLoCo rank-monotonicity sweep on Qwen3-4B and Qwen3-14B. ΔPPL = quantized − BF16 reference (lower is better), mean ± std over n=3 random sign-init seeds. Same recipe as the headline tables (greedy peel, SmoothQuant presmooth, NVFP4 mse scale rule, 128-sample C4-train calibration); only the binary rank varies. Mean ΔPPL trends down with rank on every column; small non-monotonic ticks (marked ↑) are within the n=3 sign-init noise floor (std ≤ 0.08 for Qwen3-14B, ≤ 0.58 for Qwen3-4B).

	Qwen3-4B		Qwen3-14B	
r	Wiki2	C4	Wiki2	C4
8	$+1.12_{\pm 0.56}$	$+0.68_{\pm 0.19}$	$+0.44_{\pm 0.14}$	$+0.34_{\pm 0.03}$
16	$+0.89_{\pm 0.22}$	$+0.80_{\pm 0.05}^{\uparrow}$	$+0.39_{\pm 0.02}$	$+0.35_{\pm 0.03}^{\uparrow}$
32	$+1.13_{\pm 0.22}^{\uparrow}$	$+0.67_{\pm 0.16}$	$+0.39_{\pm 0.05}$	$+0.30_{\pm 0.03}$
64	$+0.78_{\pm 0.27}$	$+0.65_{\pm 0.12}$	$+0.38_{\pm 0.07}$	$+0.32_{\pm 0.04}^{\uparrow}$
128	$+0.97_{\pm 0.41}^{\uparrow}$	$+0.64_{\pm 0.06}$	$+0.33_{\pm 0.08}$	$+0.28_{\pm 0.04}$
256	$+0.70_{\pm 0.11}$	$+0.57_{\pm 0.02}$	$+0.34_{\pm 0.05}^{\uparrow}$	$+0.29_{\pm 0.05}^{\uparrow}$
512	$+0.76_{\pm 0.23}^{\uparrow}$	$+0.63_{\pm 0.09}^{\uparrow}$	$+0.33_{\pm 0.07}$	$+0.28_{\pm 0.04}$
1024	$\mathbf{+0.58}_{\pm 0.12}$	$\mathbf{+0.46}_{\pm 0.10}$	$\mathbf{+0.27}_{\pm 0.05}$	$\mathbf{+0.21}_{\pm 0.01}$
net r=8→1024	−0.54	−0.22	−0.17	−0.13

to 0.58 for Qwen3-4B, and its mean ΔPPL drops from +0.44 to +0.27 on WikiText-2 and from +0.34 to +0.21 on C4.

A.4. BiLoCo Kernel Breakdown

Table 8 expands the B200 sweep behind Table 5. Both U and V are stored as sign bitsets. The branch keeps a persistent FP32 accumulator and writes the per-token delta $Y_{\text{acc}} - Y_{\text{prev}}$, avoiding the per-token clear required by the earlier explicit-clear version. The *join/no-add* column is the safe two-stream graph after FP4 and BiLoCo have both finished but before the final output combine. The branch column is the measured chain $xV \rightarrow zU$: the first kernel produces the small FP32 rank vector z, and the second rank-split kernel expands z through packed U into the persistent accumulator. The isolated T_{xV} and T_{zU} columns are graph-level component spans measured for attribution; they include fixed graph/launch and occupancy floors and therefore should not be added together.

The attribution is stable across ranks. The packed branch is below the native FP4 GEMM on every row, and the join/no-add time matches FP4-only within timing noise. The observed overhead is therefore the post-join tail: 1.95–2.11 μs, nearly all of $T_{\text{full}} - T_{\text{FP4}}$. The branch time grows from 9.38 μs at r=16 to 12.93 μs at r=512, but this growth remains hidden because the FP4 stream takes 15.39–15.65 μs. For the matched-storage r=512 row, the logical schedule is:

This mirrors the systems lesson from SVDQuant/Nunchaku: their engine reduces low-rank overhead by fusing the down projection with activation quantization and the up projection with low-bit output generation. Our branch arithmetic is already hidden; the analogous remaining latency target is to fold the final combine into the FP4 epilogue/mainloop without serializing the rank-parallel BiLoCo branch.

Table 8. Packed BiLoCo B200 decode breakdown at decode-5k. T_{xV} and T_{zU} are isolated graph spans and are not additive; T_{branch} is the measured packed branch chain. Times are in μs.

rank	T_{xV}	T_{zU}	T_{branch}	FP4	join/no-add	full	full/FP4	tail
16	9.28	9.44	9.38	15.55	15.65	17.66	1.136×	2.02
32	9.31	9.38	9.57	15.39	15.65	17.70	1.150×	2.05
64	9.41	9.41	9.47	15.65	15.55	17.66	1.129×	2.11
128	9.41	9.28	11.23	15.62	15.55	17.66	1.131×	2.11
256	9.44	9.38	11.33	15.58	15.65	17.70	1.136×	2.05
512	9.38	9.31	12.93	15.65	15.71	17.66	1.129×	1.95

Table 9. Logical r=512 packed-kernel timeline.

operation	stream	interval (μs)
native FP4 residual GEMM	tensor-core stream	$[0,\ 15.65]$
BiLoCo branch (xV then packed zU delta)	CUDA stream	$[0,\ 12.93]$
streams joined, no output combine	root stream	15.71
final BF16 output combine	root stream	$[15.71,\ 17.66]$

A.5. Naive SVDQuant Runtime References

Table 10 reports the dense-BF16 references used to contextualize Table 5. These are not the official SVDQuant/Nunchaku kernel. Both use the same CUDA-Graph, two-stream schedule as the BiLoCo measurement: native FP4 residual GEMM on one stream, a dense BF16 low-rank branch on a second stream, then the same BF16 output combine:

$$Y_{\text{branch}} = (XV_{\text{bf16}}^{\top})U_{\text{scaled}}.$$

For storage budget b, the SVDQuant-style rank is $r = b/16$, matching the paper's BF16-vector storage accounting. The BF16-GEMM reference computes the branch with two dense BF16 GEMMs. The CUDA-core reference uses the repository fused branch diagnostic, which recomputes the down-projection for each output tile and is therefore a conservative implementation check rather than the fair BF16 branch baseline.

Table 10. Naive SVDQuant-style two-stream references on B200 at decode-5k. The shared same-run FP4 baseline is 15.52 μs. Times are CUDA-Graph medians in μs.

b	r	BF16 branch	BF16 join	BF16 full	BF16/FP4	CUDA branch	CUDA join	CUDA full	CUDA/FP4
16	1	9.02	17.54	19.62	1.264×	8.86	17.44	19.74	1.272×
32	2	10.50	17.54	19.58	1.262×	9.06	17.28	19.58	1.262×
64	4	10.62	17.50	19.58	1.262×	9.09	17.47	19.55	1.260×
128	8	11.20	17.12	19.58	1.262×	11.04	17.50	19.58	1.262×
256	16	11.62	17.50	19.62	1.264×	13.44	17.82	21.66	1.396×
512	32	12.08	17.50	19.55	1.260×	17.54	23.42	25.60	1.649×

The BF16-GEMM row resolves the timing ambiguity: the BF16 branch itself is not the 17 μs component in the matched b=512 setting. With two BF16 GEMMs it is 12.08 μs and hides under the native FP4 stream; the remaining cost is again the output-combine tail. The slower CUDA-core b=512 row is included only to show why a naive single branch diagnostic should not be cited as the SVDQuant branch speed: it recomputes $XV^{\top}$ per output tile, causing the join/no-add time to rise to 23.42 μs.

For attribution, Table 11 keeps the older expanded-U diagnostic out of the main text. It stores V as packed signs but materializes U as BF16 signs for the branch expansion, so it is not a storage-realistic deployment result. Its role is only to show that post-join output combination, rather than signed arithmetic, was already the bottleneck in the earlier expanded-U path.

Negative packed-kernel attempts were also informative. Fixed rank-group and column-tile variants for packed zU did not improve the r=512 schedule. Cooperative-grid synchronization made the branch substantially slower, and split seed/add/finalize variants added enough launch and wave overhead to erase the no-clear benefit. The persistent delta design is therefore the current kernel path; FP4 epilogue or mainloop fusion is the remaining way to remove the final combine tail.

A.6. Full Downstream Task Table

Standard five-task battery. Table 12 reports the per-task PTQ evaluation that complements the four-model downstream table in the main text. Qwen3-32B is kept in Table 4; the standard five-task BF16 and FP4-only baselines were not run for that model. ARC-easy, ARC-challenge, PIQA, and OBQA report `acc_norm`; WinoGrande reports `acc`. All rows use C4-train calibration with SmoothQuant

Table 11. Expanded-U B200 diagnostic at decode-5k, r=512.

scope	measurement	time (μs)	interpretation
schedule	FP4 only	15.55	baseline
schedule	overlap/no-add	15.52	branch hidden; add omitted
schedule	full expanded-U	19.62	explicit post-join combine
span	$xV_{\text{packed}}^{\top}$ projection	2.59	branch stream
span	zU_{bf16} expansion	2.78	branch stream
span	branch chain	5.50	projection + expansion
span	explicit final add	1.76	post-join
span	stream-drain tail	2.94	post-join

presmoothing α=0.5 and the `mse` NVFP4 scale rule. For the three models below, BiLoCo r=512 wins the five-task mean on Qwen3-4B and Qwen3-14B and trails Llama-3-8B by 0.003. BiLoCo r=128 is included as a lower-storage reference.

Table 12. Standard PTQ downstream task breakdown. Higher is better.

Model	Method	b	ARC-e	ARC-c	PIQA	WinoG.	OBQA	mean
Qwen3-4B	SVDQuant k=32	512	0.775	0.467	**0.738**	0.639	0.296	0.583
	BiLoCo r=128	128	**0.789**	0.485	0.727	0.618	**0.298**	0.583
	BiLoCo r=512	512	0.782	**0.486**	0.724	**0.652**	0.286	**0.586**
Qwen3-14B	SVDQuant k=32	512	**0.830**	0.557	0.780	**0.717**	0.344	0.646
	BiLoCo r=128	128	**0.830**	0.560	**0.795**	0.688	0.344	0.643
	BiLoCo r=512	512	0.828	**0.583**	**0.795**	0.715	**0.348**	**0.654**
Llama-3-8B	SVDQuant k=32	512	0.787	0.472	0.775	0.716	0.328	0.616
	BiLoCo r=128	128	**0.790**	**0.485**	**0.777**	**0.728**	**0.338**	**0.624**
	BiLoCo r=512	512	0.767	0.469	0.774	**0.728**	0.326	0.613

Stderr $\leq$ 0.022 per cell. Bold marks the best value within each model block, with ties bolded.

A.7. Full KL Table

KL divergence corroborates the high-storage win. Beyond perplexity and zero-shot multiple-choice, we measure the mean and 99-th-percentile per-token Kullback–Leibler divergence $\mathrm{KL}(p_{\mathrm{BF16}} \,\|\, p_{\mathrm{quant}})$ between the BF16 reference distribution and each quantized model on a 200-sample held-out Fineweb subset (Table 13). This appendix table reports the three small/medium models from the original sweep; the main table adds Qwen3-32B. At the headline b=512 tier, BiLoCo wins mean KL on all three appendix models (Qwen3-4B 0.074 vs. 0.084, Qwen3-14B 0.048 vs. 0.052, Llama-3-8B 0.076 vs. 0.082), and also wins the worst-case KL_{p99} across all three. At b=128 the picture is mixed: BiLoCo wins Llama-3-8B and ties Qwen3-14B, while losing narrowly on Qwen3-4B. Combined with PPL and downstream, the b=512 picture is strong but not absolute: BiLoCo wins all appendix KL cells, has no mean PPL losses, and is competitive on downstream means.

Table 13. Appendix KL sweep at b=128 and b=512 on held-out Fineweb. Lower is better.

			KL mean (↓)		KL p99 (↓)	
Model	Method	b	BiLoCo	SVDQ	BiLoCo	SVDQ
Qwen3-4B	r=128 vs. r=8	128	0.0914	**0.0901**	**0.1849**	0.1965
	r=512 vs. r=32	512	**0.0740**	0.0835	**0.1421**	0.1649
Qwen3-14B	r=128 vs. r=8	128	0.0548	**0.0541**	0.2026	**0.1618**
	r=512 vs. r=32	512	**0.0480**	0.0517	**0.1303**	0.1671
Llama-3-8B	r=128 vs. r=8	128	**0.0856**	0.0881	**0.3611**	0.4789
	r=512 vs. r=32	512	**0.0756**	0.0824	**0.3252**	0.3409

Table 14. Diagnostic kernel optimization timeline for $r = 32$ at decode-5k. These are development ablations, not headline timings.

Pass	diagnostic time (μs)	vs FP4
0. Original chain (xV_S $\rightarrow$ ZU_accum)	19.6	1.46×
1. `binary_xVSU_split_n` (B_N=512, 10 CTAs)	18.0	1.34×
2. + `__launch_bounds__(NTHR, 1)` (occupancy hint)	17.4	1.30×
3. `binary_xVSU_split_rank`, r_g=8 (4 groups)	16.0	1.19×
4. Cooperative-grid finalize (`cg::this_grid().sync`)	16.0	1.19×
5. r_g=4 (8 groups, 32 concurrent CTAs)	**14.2**	**1.05**×

A.8. Kernel optimization timeline

The decisive win was Pass 5: halving the rank-group size from 8 to 4 doubled the concurrent CTA count (32 vs 16) and halved the per-CTA K-reduction work. The remaining gap in the correct schedule comes from the explicit post-add and stream synchronization.

B. Proofs

We include the formal statements and proof sketches for the signed-cut claims referenced in Section 3.3.

B.1. Monotone single-term sign alternation

Theorem B.1 (Monotone single-term sign alternation) *Run Algorithm 1's inner loop from any sign init $v^{(0)} \in \{\pm 1\}^n$, alternate $u^{(t)} = \mathrm{sign}(Rv^{(t-1)})$ and $v^{(t)} = \mathrm{sign}(R^\top u^{(t)})$ with deterministic tie-breaking, and set $\alpha^\star = (u^\top Rv)/(mn)$ where u, v are the final iterates. Then $u^\top Rv$ is non-decreasing after each coordinate update. The deflated residual $R' = R - \alpha^\star uv^\top$ satisfies*

$$\|R\|_F^2 - \|R'\|_F^2 \;=\; \frac{(u^\top Rv)^2}{mn}.$$

Monotonicity. For fixed $u \in \{\pm 1\}^m$, the v-update $v \leftarrow \mathrm{sign}(R^\top u)$ maximizes $u^\top Rv$ over $v \in \{\pm 1\}^n$ (sign-of-each-coordinate is a separable maximization). Symmetrically, the u-update maximizes $u^\top Rv$ over $u \in \{\pm 1\}^m$. Thus each coordinate update is non-decreasing in the single-term objective. With deterministic tie-breaking, the finite state space implies convergence to a fixed point or plateau.

Frobenius decrease. For any $u \in \{\pm 1\}^m$, $v \in \{\pm 1\}^n$ and $R' = R - \alpha uv^\top$, $\|R'\|_F^2 = \|R\|_F^2 - 2\alpha\, u^\top Rv + \alpha^2 mn$. Minimizing in α gives $\alpha^\star = u^\top Rv/(mn)$ and $\|R\|_F^2 - \|R'\|_F^2 = (u^\top Rv)^2/(mn)$. □

B.2. Conditional signed-cut decay

Theorem B.2 (Conditional signed-cut decay) *Let $W \in \mathbb{R}^{m\times n}$ and run a peeling procedure for r steps. At step t, let R_t be the residual and suppose the selected signs satisfy*

$$|u_t^\top R_t v_t| \;\geq\; \rho \|R_t\|_\blacksquare \qquad \textit{for some } \rho \in (0, 1].$$

With $\alpha_t = (u_t^\top R_t v_t)/(mn)$ and $R_{t+1} = R_t - \alpha_t u_t v_t^\top$, the residuals satisfy

$$\min_{0\leq t<r} \|R_t\|_\blacksquare^2 \;\leq\; \frac{mn}{\rho^2 r}\, \|W\|_F^2.$$

Let $\Phi_t = \|R_t\|_F^2$. The oracle condition and Theorem 3.1's Frobenius identity give

$$\Phi_t - \Phi_{t+1} \;=\; \frac{(u_t^\top R_t v_t)^2}{mn} \;\geq\; \frac{\rho^2 \|R_t\|_\blacksquare^2}{mn}.$$

Since $\Phi_0 = \|W\|_F^2$ and $\Phi_r \geq 0$, telescoping yields

$$\sum_{t=0}^{r-1} \|R_t\|_\blacksquare^2 \;\leq\; \frac{mn}{\rho^2}\, \|W\|_F^2.$$

At least one of the r residuals therefore satisfies $\|R_t\|_\blacksquare^2 \leq mn\|W\|_F^2/(\rho^2 r)$. This proves the stated minimum-over-iterations bound. No monotonicity of the final cut norm is required or claimed. □

B.3. Optimal coefficient refit

Theorem B.3 (Optimal α refit) *Given fixed binary $\{u_i, v_i\}_{i=1}^r$, the closed-form solution $\boldsymbol{\alpha}^* = A^{-1}b$ with $A_{ij} = (u_i^\top u_j)(v_i^\top v_j)$ and $b_i = u_i^\top W v_i$ is the unique minimizer of $\Phi(\boldsymbol{\alpha}) = \|W - \sum_i \alpha_i u_i v_i^\top\|_F^2$, provided A is invertible. The matrix A is positive semi-definite and is strictly positive definite whenever the terms are linearly independent in $\mathbb{R}^{m\times n}$.*

Let $a_i = \mathrm{vec}(u_i v_i^\top)$ and let $G = [a_1, \ldots, a_r] \in \mathbb{R}^{mn\times r}$. Then

$$\Phi(\boldsymbol{\alpha}) = \|\mathrm{vec}(W) - G\boldsymbol{\alpha}\|_2^2.$$

The normal equations are

$$G^\top G\boldsymbol{\alpha} = G^\top \mathrm{vec}(W).$$

Their entries are

$$(G^\top G)_{ij} = \langle u_i v_i^\top, u_j v_j^\top \rangle_F = (u_i^\top u_j)(v_i^\top v_j)$$

and

$$(G^\top \mathrm{vec}(W))_i = u_i^\top W v_i.$$

Thus $A = G^\top G$ and $b = G^\top \mathrm{vec}(W)$, giving $A\boldsymbol{\alpha} = b$. The Gram form makes A positive semi-definite; if the terms are linearly independent, G has full column rank and A is positive definite, so the minimizer is unique. With a numerical ridge, the implemented solve is $(A + \epsilon I)^{-1}b$. $\square$