SIGNMUON: A COMMUNICATION-EFFICIENT DISTRIBUTED MUON OPTIMIZATION FOR LLM PRE-TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Full-precision gradient exchange can limit data-parallel training. We propose SignMuon, a distributed optimizer that combines Muon's matrix update directions with signSGD's majority vote. Each worker approximates the polar factor of its local momentum matrix and sends the entrywise signs. Workers aggregate them through int8 AllReduce or packed 1-bit AllGather and apply the same update. We prove conditional stationarity bounds for the resulting vote and separate polar-solver error from the loss introduced by signing. The cubic solver bound allows arbitrarily small positive singular values; the optimizer bounds require explicit vote-alignment assumptions. We evaluate three language-model architectures with up to 1.56B parameters on eight GPUs. SignMuon attains the lowest validation log-perplexity among the tested signed optimizers at the last shared token count and elapsed time in each pretraining comparison. These results concern the recorded learning rates and training budgets. Our communication analysis accounts for both the size of each packed buffer and its replication across workers.

1 INTRODUCTION

Synchronous data-parallel training alternates local gradient computation with communication across workers. As models grow, exchanging full-precision gradients or updates can consume a substantial part of each iteration. An optimizer therefore affects training time through both the updates it computes and the information it exchanges. Reducing message size helps only if the resulting updates still make useful progress.

SignSGD addresses communication by replacing each gradient coordinate with its sign and combining workers' signs through majority vote (Bernstein et al., 2018). A worker can encode each sign in one bit. Under suitable noise assumptions, the majority-vote bound reduces the stochastic term by $1/\sqrt{M}$, where M is the worker count. The update, however, treats each coordinate separately. It does not directly use the matrix structure of weight tensors.

Muon takes a different approach to the update direction. It forms a momentum matrix and approximates its polar factor through Newton–Schulz (NS) iterations (Jordan, 2024; Shen et al., 2025). This operation uses the matrix structure of the parameters and controls the singular values of the direction. Full-precision distributed implementations still exchange floating-point information, so the local update rule alone does not remove the communication bottleneck.

We combine these operations in SignMuon. Each worker computes a polar direction from its local momentum, takes its entrywise signs, and participates in a majority vote. All workers apply the resulting sign matrix. The polar computation requires no messages: workers perform it before exchanging signs. We analyze the signed vote, with normalization for the stationarity bounds.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Algorithm 1 SIGNMUON (single worker)

Input: stepsizes $\{\eta_t\}$, momentum $\beta \in [0, 1)$, weight decay $\lambda \geq 0$, NS iterations K, stability constant $\varepsilon > 0$, scaling SCALE $\in \{\text{SPECTRAL}, \text{FRO}\}$, power iterations P.
Initialize momentum: $M_0 \leftarrow 0$.
for $t = 0, 1, \ldots, T-1$ **do**
  Draw mini-batch; compute stochastic gradient G_t at W_t.
  Apply weight decay: $\widetilde{G}_t \leftarrow G_t + \lambda W_t$.
  Update momentum: $M_{t+1} \leftarrow \beta M_t + (1-\beta)\widetilde{G}_t$.
  Compute polar direction: $U_t \leftarrow \mathrm{PolarNS}(M_{t+1}; K, \varepsilon, \text{SCALE}, P)$.
  Extract entrywise sign: $S_t \leftarrow \mathrm{sign}(U_t)$.
  Update parameters: $W_{t+1} \leftarrow W_t - \eta_t S_t$.
end for

Algorithms 1–3: local SignMuon update. SignMuon updates momentum, approximates its polar factor with POLARNS, and takes an entrywise signed step. POLARNS scales the input and applies Newton–Schulz iterations; with spectral scaling, it calls POWERITER to estimate the spectral norm. All three routines run locally, without communication.

Algorithm 2 POLARNS: Newton-Schulz polar-factor approximation

Input: $X \in \mathbb{R}^{r \times c}$, iterations K, $\varepsilon > 0$, SCALE $\in \{\text{SPECTRAL}, \text{FRO}\}$, power iterations P.
if SCALE $=$ SPECTRAL **then**
  Estimate $\|X\|_{\mathrm{op}}$: $\sigma \leftarrow \mathrm{PowerIter}(X, P, \varepsilon)$
else
  $\sigma \leftarrow \|X\|_F$
end if
$Y \leftarrow X / \max(\sigma, \varepsilon)$
$I \leftarrow I_c$ ($c \times c$ identity)
for $k = 1, \ldots, K$ **do**
  $Y \leftarrow \frac{1}{2} Y (3I - Y^\top Y)$
end for
Return Y

Algorithm 3 POWERITER: spectral-norm estimate

Input: $X \in \mathbb{R}^{r \times c}$, steps P, $\varepsilon > 0$.
Sample $v \in \mathbb{R}^c$; normalize: $v \leftarrow v / (\|v\|_2 + \varepsilon)$
for $p = 1, \ldots, P$ **do**
  $u \leftarrow Xv, \quad u \leftarrow u / (\|u\|_2 + \varepsilon)$
  $v \leftarrow X^\top u, \quad v \leftarrow v / (\|v\|_2 + \varepsilon)$
end for
Return $\|Xv\|_2$

The representation and the collective both determine communication cost. An int8 AllReduce sends one byte per sign entry and sums the signs within the collective. A packed AllGather sends one bit per entry but gives every worker all of the other workers' buffers, after which each worker computes the vote locally. The packed buffer is eight times smaller, yet AllGather replicates it across workers. We analyze these costs separately instead of equating a one-bit representation with a fixed reduction in network traffic.

Our analysis concerns the implemented momentum–polar–sign pipeline. Normalizing the vote bounds its spectral norm, while an explicit alignment condition controls progress. We also give a conditional fixed-batch rate by tracking momentum error. The local polar solver admits an inner-product error bound without a lower bound on positive singular values. A separate signing-loss term connects polar approximation to the explicit alignment assumptions used in our stationarity guarantees.

Across three architectures and three pretraining corpora on eight GPUs, SignMuon reaches the lowest mean endpoint log-perplexity among the tested sign-based optimizers within the recorded 30-minute training window. Token-efficiency and wall-clock curves describe progress under the reported configurations.

We describe the update and analyze its stationarity under explicit sign-error assumptions, then compare pretraining loss against tokens and elapsed time. The appendices give implementation details, experimental protocols, and proofs.

Algorithm 4 SIGNMUON (distributed with packed AllGather)

Input: Learning rates $\{\eta_t\}$, M workers, momentum $\beta \in [0, 1)$, weight decay $\lambda \geq 0$, NS iterations K, tolerance $\varepsilon > 0$, SCALE $\in \{\text{SPECTRAL}, \text{FRO}\}$, power iterations P.
Initialize each worker's momentum: $M_0^{(m)} \leftarrow 0, \quad m = 1, \ldots, M$.
for $t = 0, 1, \ldots, T-1$ **do**
  Each worker m **independently:**
  Compute local gradient $G_t^{(m)}$ at W_t.
  Apply weight decay: $\widetilde{G}_t^{(m)} \leftarrow G_t^{(m)} + \lambda W_t$.
  Update local moment: $M_{t+1}^{(m)} \leftarrow \beta M_t^{(m)} + (1-\beta)\widetilde{G}_t^{(m)}$.
  Compute polar decomposition: $U_t^{(m)} \leftarrow \mathrm{PolarNS}\big(M_{t+1}^{(m)}; K, \varepsilon, \text{SCALE}, P\big)$.
  Extract local sign: $S_t^{(m)} \leftarrow \mathrm{sign}(U_t^{(m)}) \in \{-1, +1\}^d$.
  Pack signs (8 signs per byte): $B_t^{(m)} \leftarrow \mathrm{PackBits}(S_t^{(m)})$.
  AllGather (per buffer):
  Gather packed `uint8` buffers: $\{B_t^{(1)}, \ldots, B_t^{(M)}\} \leftarrow \mathrm{all_gather}(B_t^{(m)})$.
  Each worker m **independently:**
  Unpack signs for $m' = 1, \ldots, M$: $S_t^{(m')} \leftarrow \mathrm{UnpackBits}(B_t^{(m')})$.
  Compute majority vote (ties to $+1$): $\bar{S}_t \leftarrow \mathrm{sign}\left(\sum_{m'=1}^{M} S_t^{(m')}\right)$.
  Update parameters: $W_{t+1} \leftarrow W_t - \eta_t \bar{S}_t$.
end for
Communication per worker: send and receive $(M-1)\lceil d/8 \rceil$ bytes each; total $2(M-1)\lceil d/8 \rceil$ bytes per buffer.
Encoding: $\geq 0 \to 1$, $< 0 \to 0$. Decoding: $0 \to -1$, $1 \to +1$.

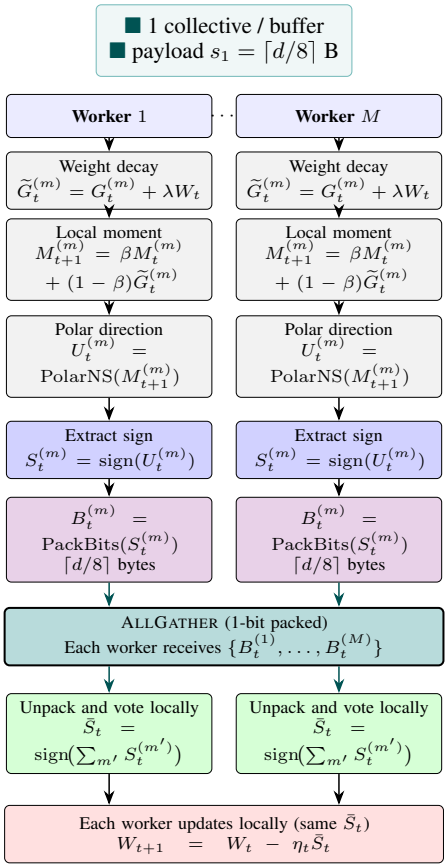

Figure 3: Schematic of distributed SIGNMUON with 1-bit ALLGATHER (Algorithm 4). Each worker computes the same local pipeline as in the AllReduce variant, then *packs* its sign vector into one bit per entry ($\lceil d/8 \rceil$ bytes). A single ALLGATHER disseminates every worker's packed buffer to every other worker. Each worker independently unpacks the gathered buffers and computes the majority vote $\bar{S}_t = \mathrm{sign}(\sum_{m'} S_t^{(m')})$ *locally*; all workers then apply the same parameter update. The diagram shows one d-entry buffer; bucketed training repeats the collective and adds scalar checks.

2 RELATED WORK

Sign methods and distributed Lion. Bernstein et al. (2018) analyze signSGD with majority vote. Lion uses signed momentum updates (Chen et al., 2023); Distributed Lion communicates binary or low-precision worker updates (Liu et al., 2024). It is a close comparator because it also combines local momentum with compressed aggregation. Jiang & Zhang (2025) establish centralized and distributed Lion rates under specified parameter schedules and analyze two-way compression, including unbiased randomized signs. Their guarantees depend on the update and compression rule and do not transfer to deterministic polar signs. Sun et al. (2023) establish small-batch convergence for momentum signSGD under weaker noise assumptions. SignMuon differs through the local polar transform, whose sign alignment needs a separate argument. Our pretraining experiments include SignLion with packed sign aggregation. Recently, OLion Wang et al. (2026), which appeared after the initial submission of this work, independently applies an entrywise sign after orthogonalization; however, it does not develop or evaluate communication-efficient distributed aggregation.

QSGD and TernGrad quantize gradients (Alistarh et al., 2017; Wen et al., 2017); Deep Gradient Compression sparsifies them (Lin et al., 2017); PowerSGD exchanges low-rank factors (Vogels et al., 2019); and error feedback compensates for compression error (Stich et al., 2018). SignMuon instead signs locally computed polar directions before majority voting.

Matrix updates and momentum. K-FAC and Shampoo construct matrix preconditioners (Martens & Grosse, 2015; Gupta et al., 2018). Muon uses momentum and polar-factor iterations (Jordan, 2024; Shen et al., 2025; Higham, 1986; Björck & Bowie, 1971). SignMuon retains this local transformation and compresses its output. Momentum (Polyak, 1964; Sutskever et al., 2013) and weight decay (Loshchilov & Hutter, 2017) remain distinct implementation choices.

Recent analyses of polar signs and inexact solvers. Qian & Richtárik (2026) analyze inexact linear minimization oracle (LMO) updates when positive singular values approach zero. Appendix F.6 relates this work to our SignMuon analysis and its alignment assumptions.

3 SignMuon

Let $W_t \in \mathbb{R}^{m\times n}$ denote a parameter block at iteration t, with $d = mn$ entries. Worker $p \in \{1, \dots, M\}$ computes a stochastic gradient $G_t^{(p)}$ from its local mini-batch. We use $\|\cdot\|_{\mathrm{op}}$, $\|\cdot\|_{\mathrm{F}}$, and $\|\cdot\|_1$ for the spectral, Frobenius, and entrywise ℓ_1 norms; $\langle A, B\rangle = \mathrm{tr}(A^\top B)$. For a compact singular value decomposition $X = U\Sigma V^\top$, we define $\mathrm{polar}(X) = UV^\top$ on the nonzero singular subspace and $\mathrm{polar}(0) = 0$.

3.1 Local direction and global vote

Worker p first updates its momentum buffer:

$$\widetilde{G}_t^{(p)} = G_t^{(p)} + \lambda W_t, \qquad M_{t+1}^{(p)} = \beta M_t^{(p)} + (1-\beta)\widetilde{G}_t^{(p)}. \tag{1}$$

Here β is the momentum coefficient and λ is the weight-decay coefficient. The worker approximates the momentum's polar factor and extracts its signs:

$$U_t^{(p)} = \mathrm{PolarNS}(M_{t+1}^{(p)}; K, \varepsilon, \textsc{scale}, P), \qquad S_t^{(p)} = \mathrm{sign}(U_t^{(p)}). \tag{2}$$

The routine uses K Newton–Schulz iterations and either Frobenius scaling or a spectral-norm estimate from P power iterations. Algorithm 2 normalizes the input, applies the cubic matrix iteration, and returns the approximate polar factor. For spectral scaling, Algorithm 3 alternates multiplication by the matrix and its transpose, normalizing each vector before returning a norm estimate. A finite power-iteration estimate need not upper-bound the spectral norm; the solver guarantee uses a valid upper bound, as provided by Frobenius scaling. We take signs entrywise and map zero to $+1$ in the binary communication algorithms, including ties in the majority vote.

Workers then compute

$$\bar{S}_t = \mathrm{sign}\left(\sum_{p=1}^{M} S_t^{(p)}\right), \qquad W_{t+1} = W_t - \eta_t \bar{S}_t. \tag{3}$$

For a single worker, $\bar{S}_t = S_t^{(1)}$. Algorithm 1 gives this local case, and Algorithm 5 gives the distributed AllReduce implementation in the appendix. Algorithm 4 and Figure 3 give the packed AllGather implementation on page 3.

3.2 Normalization and local orthogonalization

The entrywise sign operation does not preserve the singular values of $U_t^{(p)}$. We therefore distinguish the local polar direction from the final update direction. For the analysis, we use

$$D_t = \frac{\bar{S}_t}{\sqrt{mn}}, \qquad W_{t+1} = W_t - \eta_t D_t. \tag{4}$$

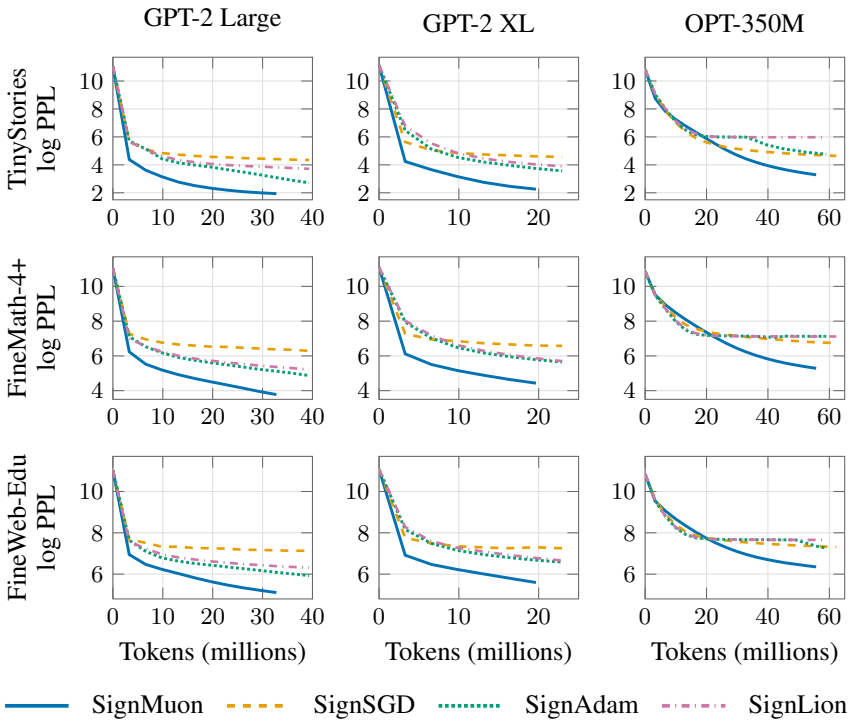

Figure 1: Single-seed pretraining token efficiency. Rows identify datasets and columns identify architectures. Curves show raw validation log-perplexity, including initialization, without running-minimum smoothing. SignMuon has the lowest loss at each panel's last shared token count. Endpoints differ across methods; matched-token comparisons use only the overlap of the curves.

Since $\|\bar{S}_t\|_{\mathrm{op}} \le \|\bar{S}_t\|_{\mathrm{F}} \le \sqrt{mn}$, this choice ensures $\|D_t\|_{\mathrm{op}} \le 1$. For one block, identical trajectories require $\eta_t^{\mathrm{theory}} = \sqrt{mn}\,\eta_t^{\mathrm{impl}}$. This conversion depends on block shape, so it does not give one global rescaling for unequal layers. Appendix F.5.2 gives a joint bound with explicit block scales.

Computing $\mathrm{polar}(\bar{S}_t)$ after the vote would require only local work, but would change the direction. Our algorithms and stationarity bounds use the signed vote directly.

3.3 Two implementations of majority vote

The int8 implementation concatenates sign tensors, performs an elementwise integer SUM AllReduce, and thresholds the resulting sums. The sum lies between $-M$ and M, so the accumulator type must represent that range. Int8 suffices for the eight-worker experiments; larger worker counts can require a wider type. All workers receive the same sum and apply the same tie rule.

Algorithm 4 and Figure 3 show the packed path. Workers start from identical parameters and zero local moments. Each computes its mini-batch gradient, adds weight decay, updates its own momentum, and applies PolarNS. It takes the entrywise sign and encodes nonnegative entries as one and negative entries as zero, packing eight signs per byte. AllGather then delivers every worker's packed buffer to every worker; workers call one sign collective per bucket and separate scalar collectives for checks and logging.

Each worker unpacks the buffers and counts positive signs coordinatewise. If c_i of the M signs are positive, their signed sum is $2c_i - M$; the majority is $+1$ when this is nonnegative and -1 otherwise. Workers discard padding bits. Finally, every worker subtracts the same learning-rate-scaled majority sign from its parameters, preserving synchronization while retaining its own momentum. Figure 3's upper boxes are local computations, its central bar is the network exchange, and its lower boxes perform local voting and updating. AllReduce gives the same vote for identical input signs, representable sums, and the same tie rule; its schematic is in Appendix A.

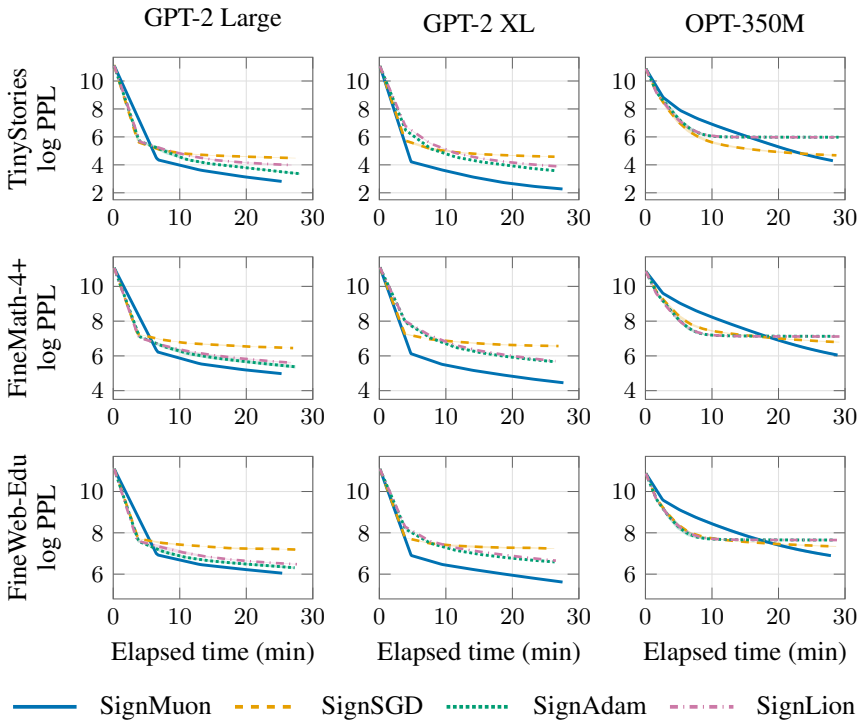

Figure 2: Multi-seed pretraining wall-clock performance. Lines show mean validation log-perplexity; bands are ± one sample standard deviation over three seeds. SignMuon is lowest at each panel's latest shared time. Interpolation uses only time ranges observed by all three seeds; no curve is extrapolated. Table 1 uses per-seed last evaluations. Timings are compared only within each architecture's campaign.

4 SignMuon Theory and Communication

4.1 Alignment of the momentum-polar vote

Let $g_t = \nabla f(W_t)$ for one matrix variable with $d = mn$ entries. Assume continuously differentiable $f(W) \geq f^* > -\infty$, fix W_0, and use deterministic $\eta_t > 0$. Spectral-norm smoothness means

$$\|\nabla f(W) - \nabla f(W')\|_* \leq L_* \|W - W'\|_{\mathrm{op}}, \tag{5}$$

where $\|\cdot\|_*$ is the nuclear norm. For SignMuon's actual vote $\bar{S}_t$ and normalized direction $D_t = \bar{S}_t/\sqrt{d}$, the descent inequality is

$$f(W_{t+1}) \leq f(W_t) - \eta_t \langle g_t, D_t \rangle + \frac{L_* \eta_t^2}{2}. \tag{6}$$

Appendix F.5.2 extends this statement to jointly updated blocks and the unnormalized implementation. Spectral smoothness alone does not improve the curvature constant for a normalized sign step (Appendix F.5.1).

Define the error probability of the final vote and its gradient-weighted loss by

$$q_{ij,t} = \Pr(\bar{S}_{t,ij} \neq \mathrm{sign}(g_{t,ij}) \mid W_t), \qquad e_t = \frac{1}{\sqrt{d}} \sum_{ij} |g_{t,ij}| q_{ij,t}. \tag{7}$$

These probabilities describe the full momentum–polar–sign pipeline, including aggregation. They are not error probabilities for signs of raw stochastic gradients. Lemma 9 gives $\mathbb{E}[\langle g_t, D_t \rangle \mid W_t] \geq V_t - 2e_t$, where $V_t = \|g_t\|_1/\sqrt{d}$.

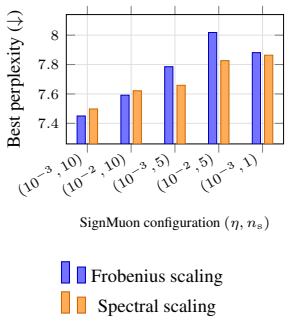

Figure 4: nanoGPT scaling comparison across learning rates η and Newton–Schulz counts n_s. Paired best perplexities differ by less than 2.5%; neither scaling wins in every configuration. Both yield very similar perplexities; we use only Frobenius scaling in the main pretraining experiments.

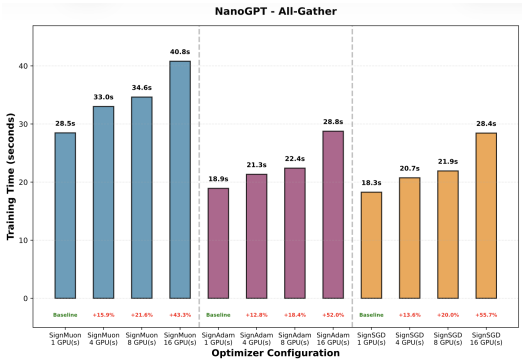

Figure 5: nanoGPT AllGather weak scaling with 1, 4, 8, and 16 GPUs. Labels report training time and the increase over each optimizer's one-GPU baseline. **SignMuon takes more time per optimizer step, but is attains best accuracy eary compared to others (Figure 2).**

Theorem 1 (Conditional SignMuon stationarity). *Let* $A_T = \sum_{t<T} \eta_t$, $B_T = \sum_{t<T} \eta_t^2$, *and* $\Delta = f(W_0) - f^*$. *Suppose the actual SignMuon vote satisfies* $2\mathbb{E}e_t \le \rho \mathbb{E}V_t + \delta_t$ *for fixed* $0 \le \rho < 1$ *and deterministic* $\delta_t \ge 0$. *Then*

$$\frac{1}{A_T} \sum_{t<T} \eta_t \mathbb{E}V_t \le \frac{1}{1-\rho} \left[\frac{\Delta}{A_T} + \frac{L_* B_T}{2A_T} + \frac{\sum_{t<T} \eta_t \delta_t}{A_T} \right]. \tag{8}$$

The proof in Appendix F sums the descent inequalities and absorbs the relative sign-error term. For $\eta_t = \eta_0/\sqrt{T}$, a residual average of order $T^{-1/2}$ gives the same stationarity order. Along an infinite run, weighted average stationarity follows if $A_T \to \infty$, $B_T/A_T \to 0$, and the weighted residual average vanishes. A nonvanishing residual leaves a bound on average stationarity. These statements concern expected gradient norms, not zero training error or convergence of the last iterate.

4.2 Momentum and Polar-Solver Error

A second result analyzes fixed-batch SignMuon directly through momentum tracking. Let $\bar{M}_{t+1} = M^{-1} \sum_p M_{t+1}^{(p)}$ and suppose the final normalized vote satisfies

$$\mathbb{E}\langle \bar{M}_{t+1}, D_t \rangle \ge \kappa \mathbb{E}\|\bar{M}_{t+1}\|_* - \varepsilon_t, \qquad 0 < \kappa \le 1. \tag{9}$$

Under conditionally unbiased gradients with bounded variance, Theorem 13 gives $T^{-1} \sum_{t<T} \mathbb{E}\|g_t\|_* = O(T^{-1/4})$ when $\eta = \eta_0 T^{-3/4}$, $\beta = 1 - T^{-1/2}$, and $T^{-1} \sum_{t<T} \varepsilon_t = O(T^{-1/4})$, with κ independent of T. This adapts the momentum-tracking argument of Qian & Richtárik (2026) to the final SignMuon direction. Appendix F.5.3 extends the rate to unnormalized blocks and biased inputs. Alignment remains additional; clipping, fixed momentum, and experimental schedules do not automatically satisfy the rate assumptions.

The local numerical error and the signing loss play different roles. For momentum X, $\|U\|_{\mathrm{op}} \le 1$, and $D = \mathrm{sign}(U)/\sqrt{d}$, let $[a]_+ = \max(a, 0)$ and define

$$\epsilon_{\mathrm{NS}} = \|X\|_* - \langle X, U \rangle, \qquad \chi_{\mathrm{sign}} = [\langle X, U - D \rangle]_+. \tag{10}$$

Proposition 12 shows $\langle g, D \rangle \ge \|g\|_* - 2\|g - X\|_* - \epsilon_{\mathrm{NS}} - \chi_{\mathrm{sign}}$. For rank-$r$ momentum and cubic Newton–Schulz scaled by $\|X\|_{\mathrm{F}}$, Theorem 11 gives $\epsilon_{\mathrm{NS}} \le r(2/3)^K \|X\|_*$, even without a lower bound on the least positive singular value. A scaling floor adds a tolerance term. These bounds control numerical inexactness, but not the loss from signing or majority voting.

4.3 Payload Size and Collective Cost

We use the latency–bandwidth model $T(s) = \alpha + \beta_{\mathrm{net}} s$ for a message of s bytes. Here β_{net} denotes inverse bandwidth, distinct from the momentum coefficient. Related communication models appear

Table 1: Combined pretraining results: mean final validation log-perplexity (sample variance), three seeds. SignMuon has the lowest mean in all nine dataset–model pairs (bold).

Optimizer	GPT-2 Large	GPT-2 XL	OPT-350M
TinyStories			
SignMuon	$\mathbf{2.7687}$ (1.0×10^{-4})	$\mathbf{2.2546}$ (7.7×10^{-5})	$\mathbf{4.2259}$ (2.7×10^{-4})
SignSGD	4.4749 (1.4×10^{-4})	4.5806 (2.2×10^{-4})	4.6765 (1.5×10^{-3})
SignAdam	3.3312 (4.4×10^{-3})	3.5693 (1.4×10^{-5})	5.9780 (5.3×10^{-9})
SignLion	3.9634 (5.1×10^{-3})	3.8535 (5.1×10^{-3})	5.9732 (2.7×10^{-8})
FineMath-4+			
SignMuon	$\mathbf{4.9500}$ (4.0×10^{-5})	$\mathbf{4.4443}$ (1.3×10^{-4})	$\mathbf{6.0110}$ (8.5×10^{-3})
SignSGD	6.4434 (3.2×10^{-4})	6.5380 (1.1×10^{-3})	6.7838 (4.1×10^{-4})
SignAdam	5.3616 (1.3×10^{-2})	5.6496 (1.4×10^{-5})	7.1280 (6.7×10^{-6})
SignLion	5.5795 (3.4×10^{-3})	5.6571 (5.8×10^{-3})	7.1168 (4.3×10^{-6})
FineWeb-Edu			
SignMuon	$\mathbf{6.0239}$ (7.4×10^{-5})	$\mathbf{5.5985}$ (2.6×10^{-5})	$\mathbf{6.8540}$ (4.9×10^{-4})
SignSGD	7.1856 (2.9×10^{-4})	7.2327 (3.0×10^{-4})	7.3340 (3.6×10^{-4})
SignAdam	6.2894 (4.9×10^{-3})	6.5859 (2.2×10^{-5})	7.6522 (9.9×10^{-8})
SignLion	6.4783 (3.6×10^{-4})	6.5896 (3.3×10^{-3})	7.6505 (2.2×10^{-7})

in distributed numerical solvers (Kumar, 2014; 2015; Kumar et al., 2013). For d communicated entries, an int8 buffer has $s_8 = d$ bytes and a packed binary buffer has $s_1 = \lceil d/8 \rceil$ bytes.

For a ring with $M > 1$ workers, the two collective costs are

$$T_{\mathrm{AR}} \approx 2(M-1)\alpha + 2(1-1/M)d\,\beta_{\mathrm{net}}, \tag{11}$$

$$T_{\mathrm{AG}} \approx (M-1)\alpha + (M-1)\lceil d/8 \rceil\,\beta_{\mathrm{net}}. \tag{12}$$

These expressions model communication, not the full training iteration. Int8 AllReduce reduces the bandwidth term by a factor of four relative to float32 AllReduce with the same collective. Packing reduces the local representation by a factor of 32 relative to float32, but AllGather replicates that representation. Ignoring byte padding, its bandwidth term divided by that of int8 AllReduce is $M/16$. Thus packed AllGather has a smaller modeled bandwidth term below 16 workers and a comparable term at 16.

AllGather also requires a buffer of $M\lceil d/8 \rceil$ bytes and local vote computation. AllReduce returns one summed buffer and avoids that replication. Neither collective communicates the polar factors. Appendix E derives the costs, including other collective schedules and parameter-server variants.

5 Experiments

5.1 Pretraining across architectures and corpora

We train GPT-2 Large, GPT-2 XL, and OPT-350M from random initialization using their native model configurations and tokenizers. We compare SignMuon, SignSGD, SignAdam, and SignLion on TinyStories, FineMath-4+, and FineWeb-Edu (`sample-10BT`).

Experimental setup. Pretraining comparisons use eight RTX 5090 or Quadro RTX 6000 GPUs, three training seeds, and a 30-minute wall-clock budget per run, including training, validation, and communication. All methods use packed AllGather and are compared within the same architecture and GPU configuration. Appendix A.6 provides the full setup. NanoGPT weak-scaling experiments hold work per GPU fixed across one to 16 RTX 4090 GPUs, while additional nanoGPT pretraining experiments use a five-minute budget. Appendix A.7 describes the normalization and weak-scaling settings; Appendix B details the nanoGPT pretraining setup.

Evaluation metrics. Validation log-perplexity is mean next-token cross-entropy in nats per token; lower is better. Figure 1 plots single-seed validation loss against cumulative training tokens processed. Figure 2 plots mean validation loss against elapsed time, with bands showing $\pm$ one sample standard deviation across three seeds. For each optimizer, we linearly interpolate the seed trajectories only

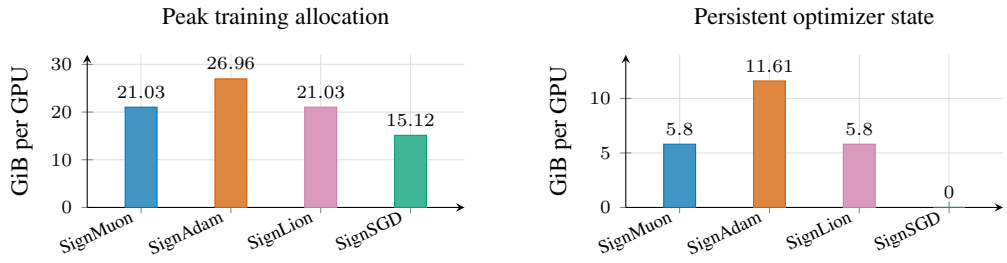

Figure 6: GPT-2 XL memory per GPU on eight RTX 5090 GPUs (TinyStories, seed 1337). SignMuon uses less peak memory than SignAdam and half its optimizer state. Peaks include temporary allocations; state counts optimizer tensors only.

within their shared observed time interval. Table 1 reports, for each method, the mean and sample variance across three seeds, using each run's final logged validation loss.

5.2 Experimental results

Token efficiency. Figure 1 shows sustained gains in token efficiency across the evaluated architectures and datasets. For GPT-2 Large and GPT-2 XL, SignMuon achieves lower validation loss than the other signed optimizers at every shared evaluation checkpoint after initialization. For OPT-350M, it establishes a consistent advantage after approximately 23 to 26 million training tokens, depending on the dataset, and maintains that advantage throughout the remaining shared token range. At the final shared token count, SignMuon achieves the lowest validation loss across all nine combinations of architecture and dataset. On TinyStories with GPT-2 XL, SignMuon achieves better validation performance with one-third as many training tokens: a loss of 3.6554 after 6.5536M tokens, compared with SignAdam's 3.7429 after 19.6608M tokens.

Performance under a fixed time budget. Under the same wall-clock budget, SignMuon achieves the lowest mean validation loss across all nine settings (Figure 2). On FineMath-4+ with GPT-2 XL, it reaches a mean validation loss of 5.7 in less than one-third of the time taken by SignAdam: approximately 8.0 minutes compared with 25.4 minutes. Table 1 further shows the lowest mean final validation loss for SignMuon in every setting. For GPT-2 XL, SignMuon also uses half the persistent optimizer memory of SignAdam (5.80 versus 11.61 GiB per GPU) and reduces peak training VRAM from 26.96 to 21.03 GiB (Figure 6). Appendix A.8 provides the memory measurement protocol and results for all three model sizes.

Sensitivity to polar normalization. In the separate nanoGPT study, Frobenius and spectral scaling yield best perplexities within 2.5% of each other across the five paired settings in Figure 4. Frobenius scaling yields lower perplexity at ten Newton–Schulz iterations, while spectral scaling yields lower perplexity at five and one. The main pretraining experiments use Frobenius scaling with ten iterations.

Distributed weak scaling. Runtime increases by 43.3% for SignMuon, compared with 52.0% for SignAdam and 55.7% for SignSGD (Figure 5). SignMuon has the smallest relative increase but a higher absolute runtime. In the additional nanoGPT pretraining experiments, it achieves the lowest mean validation loss on all three datasets (Table 7; Appendix D).

6 Conclusion

SignMuon combines local polar-direction computation with packed 1-bit majority-vote communication. Across the evaluated architectures and datasets, it improves token efficiency and achieves lower validation loss within the same wall-clock budget. These results demonstrate the value of matrix-based updates for learning efficiently while communicating only packed signs. Our theory complements these findings with conditional stationarity guarantees that separate momentum, polar-solver, and signing errors.

Reproducibility statement. Algorithms 1 through 4 specify the local optimization and distributed aggregation routines. Appendix A.6 documents data preparation, optimizer hyperparameters, numerical precision, training seeds, and evaluation procedures. Appendix B gives the separate nanoGPT experimental protocol. Appendices E and F provide the communication analysis and proofs of the theoretical results. We will make the training code publicly available after the review process concludes.

AI use statement. We used generative AI to refine the prose, improve clarity and readability, assist with the organization and presentation of the paper, and provide coding assistance. We take responsibility for the final manuscript, including its theoretical claims and experimental results.

REFERENCES

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, 2017.

Jeremy Bernstein, Jiawei Zhao, Kamyar Azizzadenesheli, and Anima Anandkumar. signSGD with majority vote is communication efficient and fault tolerant. *CoRR*, abs/1810.05291, 2018. URL https://arxiv.org/abs/1810.05291.

Åke Björck and Clyde Bowie. An iterative algorithm for computing the best estimate of an orthogonal matrix. *SIAM Journal on Numerical Analysis*, 8(2):358–364, 1971. doi: 10.1137/0708036.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms. In *Advances in Neural Information Processing Systems 36*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/9a39b4925e35cf447ccba8757137d84f-Abstract-Conference.html.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization, 2018. URL https://arxiv.org/abs/1802.09568. arXiv:1802.09568.

Nicholas J. Higham. Computing the polar decomposition—with applications. *SIAM Journal on Scientific and Statistical Computing*, 7(4):1160–1174, 1986.

Torsten Hoefler, Timo Schneider, and Andrew Lumsdaine. Characterizing the influence of system noise on large-scale applications by simulation. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pp. 1–11, 2010.

Wei Jiang and Lijun Zhang. Convergence analysis of the lion optimizer in centralized and distributed settings. *arXiv preprint arXiv:2508.12327*, 2025. URL https://arxiv.org/abs/2508.12327.

Keller Jordan. Muon: An optimizer for hidden layers in neural networks. https://kellerjordan.github.io/posts/muon/, 2024. Blog post. Accessed: 2025-12-26.

Pawan Kumar. Communication optimal least squares solver. In *2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC,CSS,ICESS)*, pp. 316–319, 2014. doi: 10.1109/HPCC.2014.55.

Pawan Kumar. Multilevel communication optimal least squares. *Procedia Computer Science*, 51:1838–1847, 2015. ISSN 1877-0509. doi: 10.1016/j.procs.2015.05.410. URL https://www.sciencedirect.com/science/article/pii/S1877050915012181. International Conference On Computational Science, ICCS 2015.

Pawan Kumar, Stefano Markidis, Giovanni Lapenta, Karl Meerbergen, and Dirk Roose. High performance solvers for implicit particle in cell simulation. *Procedia Computer Science*, 18:2251–2258, 2013. ISSN 1877-0509. doi: 10.1016/j.procs.2013.05.396. URL https://www.sciencedirect.com/science/article/pii/S1877050913005395. 2013 International Conference on Computational Science.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J. Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *CoRR*, abs/1712.01887, 2017. URL https://arxiv.org/abs/1712.01887.

Bo Liu, Lemeng Wu, Lizhang Chen, Kaizhao Liang, Jiaxu Zhu, Chen Liang, Raghuraman Krishnamoorthi, and Qiang Liu. Communication efficient distributed training with distributed lion. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/20cea6c1b36ae5f69c48427a68b67fbc-Abstract-Conference.html.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *CoRR*, abs/1711.05101, 2017. URL https://arxiv.org/abs/1711.05101.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2408–2417. PMLR, 2015.

B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964. doi: 10.1016/0041-5553(64)90137-5.

Xun Qian and Peter Richtárik. Convergence analysis of Muon-type methods with inexact LMO in the degenerate case. *arXiv preprint arXiv:2606.21581*, 2026. URL https://arxiv.org/abs/2606.21581.

Rolf Rabenseifner. Optimization of collective reduction operations. In *International Conference on Computational Science (ICCS)*, pp. 1–9, 2004.

Wei Shen, Ruichuan Huang, Minhui Huang, Cong Shen, and Jiawei Zhang. On the convergence analysis of muon, 2025. URL https://arxiv.org/abs/2505.23737. arXiv:2505.23737.

Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. *Advances in neural information processing systems*, 31, 2018.

Tao Sun, Qingsong Wang, Dongsheng Li, and Bao Wang. Momentum ensures convergence of SIGNSGD under weaker assumptions. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 33077–33099. PMLR, 2023. URL https://proceedings.mlr.press/v202/sun23l.html.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1139–1147. PMLR, 2013.

Rajeev Thakur, Rolf Rabenseifner, and William Gropp. Optimization of collective communication operations in MPICH. *International Journal of High Performance Computing Applications*, 19(1):49–66, 2005.

Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. PowerSGD: Practical low-rank gradient compression for distributed optimization. In *Advances in Neural Information Processing Systems*, 2019.

Zixiao Wang, Yifei Shen, and Huishuai Zhang. Olion: Approaching the hadamard ideal by intersecting spectral and ℓ_∞ implicit biases, 2026. URL https://arxiv.org/abs/2602.01105.

Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. *Advances in neural information processing systems*, 30, 2017.

A Implementation details

The main text gives the single-worker and packed AllGather algorithms, the local polar and norm-estimation routines, and the packed communication schematic. This appendix gives the int8 AllReduce algorithm and schematic, with further implementation details. All workers start from the same parameters and apply the same aggregate, while retaining separate local momentum buffers.

Algorithm 5 SIGNMUON (distributed with all-reduce)

Input: Learning rates $\{\eta_t\}$, M workers, momentum $\beta \in [0, 1)$, weight decay $\lambda \geq 0$, NS iterations K, tolerance $\varepsilon > 0$, scale $\in \{\text{SPECTRAL}, \text{FRO}\}$, power iterations P
Initialize: $M_0^{(m)} \leftarrow 0$ for all workers $m \in \{1, \ldots, M\}$
for $t = 0, 1, \ldots, T-1$ **do**
  Each worker m **independently:**
    Compute local gradient $G_t^{(m)}$ at current parameters W_t
    Apply weight decay: $\widetilde{G}_t^{(m)} \leftarrow G_t^{(m)} + \lambda W_t$
    Update local momentum: $M_{t+1}^{(m)} \leftarrow \beta M_t^{(m)} + (1-\beta)\widetilde{G}_t^{(m)}$
    Compute polar decomposition: $U_t^{(m)} \leftarrow \mathrm{PolarNS}(M_{t+1}^{(m)}; K, \varepsilon, \text{SCALE}, P)$
    Extract local sign: $S_t^{(m)} \leftarrow \mathrm{sign}(U_t^{(m)}) \in \{-1, +1\}^d$
  All-reduce (collective):
    All workers participate in sum-reduce:
      $\text{sum_signs} \leftarrow \sum_{m=1}^{M} S_t^{(m)}$ *(all-reduce with SUM, int8)*
    All workers compute majority vote locally:
      $\bar{S}_t \leftarrow \mathrm{sign}(\text{sum_signs}) \in \{-1, +1\}^d$ *(ties default to +1)*
  Each worker m **independently:**
    Update parameters: $W_{t+1} \leftarrow W_t - \eta_t \bar{S}_t$
end for
Local payload: d bytes (int8). Appendix E gives total traffic.

A.1 ALLREDUCE IMPLEMENTATION

Algorithm 5 gives the integer-AllReduce implementation illustrated in Figure 7. Each worker computes its gradient, momentum, and polar direction locally, then contributes an integer sign buffer to a single sum reduction. Thresholding the returned sums produces the same majority vote on every worker, so the parameter update needs no further communication. Unlike the packed AllGather path in Algorithm 4, this implementation receives one aggregate buffer rather than every worker's input; the accumulator must represent all possible sums without overflow.

A.2 PACKED MAJORITY VOTE

Algorithm 4 and Figure 3 implement majority voting through packed AllGather. The encoding below explains how each worker recovers the same integer sign sum locally from the received bits, including the tie rule and the treatment of padding.

The displayed algorithms treat one matrix or concatenated buffer. The recorded implementation uses one sign AllGather per bucket and separate scalar collectives for checks and reporting (Appendix A.6). Each worker encodes a nonnegative entry as one and a negative entry as zero, packs eight entries per byte, and AllGathers the packed buffer. It then counts the positive signs for each coordinate. If that count is c_i, the signed sum is $2c_i - M$. The worker applies the sign of this sum, with ties mapped to $+1$. Byte padding does not represent additional parameter entries and must not enter the vote.

A.3 LOCAL COMPUTATION COST

For an $r \times c$ matrix, Algorithm 2 forms $Y^\top Y$ and multiplies by that $c \times c$ Gram matrix. Each Newton–Schulz iteration costs $O(rc^2)$ arithmetic operations with classical dense multiplication. With K such iterations and P power iterations, the local cost is $O(Krc^2 + Prc)$; extracting signs costs $O(rc)$. The Gram matrix needs $O(c^2)$ temporary storage in addition to matrix-sized buffers. A thin SVD costs $O(rc \min\{r, c\})$. Applying Newton–Schulz to the transpose when $c > r$ can reduce its cost to $O(Krc \min\{r, c\})$, but the displayed routine uses the right Gram matrix as written. GPU runtime also depends on the hardware and kernel implementation.

SignMuon therefore adds local matrix products relative to signSGD or Lion. Compression helps end-to-end time only when communication savings and optimization progress compensate for that work. Persistent optimizer state excludes temporary Gram and multiplication buffers. Figure 6 and

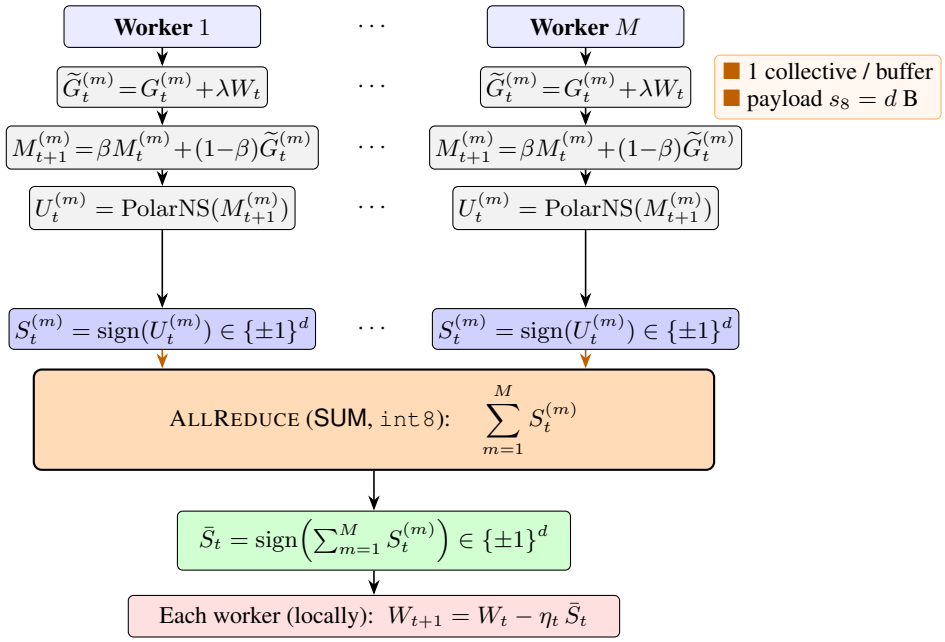

Figure 7: Schematic of distributed SignMuon with SUM AllReduce (Algorithm 5). Each worker independently computes its momentum $M_{t+1}^{(m)}$, the Newton–Schulz polar direction $U_t^{(m)}$, and the entrywise sign $S_t^{(m)} \in \{-1, +1\}^d$. A single integer SUM AllReduce aggregates the sign buffers across workers; each worker then thresholds locally with $\mathrm{sign}(\cdot)$ to recover the majority vote $\bar{S}_t$, and applies the parameter update with no further communication. Normalization and Newton–Schulz are local. For identical input signs and no overflow, this one-buffer collective gives the same vote as packed AllGather; we exclude auxiliary scalar collectives.

Table 4 distinguish this stored state from peak training allocation, which also includes temporary workspace.

A.4 Local polar approximation

Algorithm 2 gives the PolarNS routine: it first scales its input, then applies the cubic Newton–Schulz iteration. Algorithm 3 supplies the optional spectral-norm estimate. Frobenius scaling uses $\|X\|_{\mathrm{F}}$ directly. Spectral scaling estimates $\|X\|_{\mathrm{op}}$ through power iteration. The estimate need not upper-bound the true norm, so the idealized convergence condition in Theorem 11 does not follow from a finite power-iteration budget alone. The numerical tolerance prevents division by zero.

A.5 Communication diagrams

Figure 7 shows the int8 path. The collective sums the local signs, and each worker thresholds the result. Figure 3 shows the packed path, where every worker receives all packed buffers and performs the reduction locally. Both paths keep the momentum and polar computations on each worker. In both diagrams, arrows entering the collective carry signs rather than gradients or polar factors. The lower boxes show the shared update after aggregation; differences in the collective affect buffer storage and network traffic, as summarized in Table 8.

A.6 Pretraining setup and provenance

Models and hardware. GPT-2 Large, GPT-2 XL, and OPT-350M are trained from random initialization using their native Hugging Face configurations, with all parameters trainable. Table 2 lists their dimensions and parameter counts. OPT-350M retains its 512-dimensional word-embedding projection and native positional capacity of 2,048; all experiments use a context length of 1,024.

Table 2: Native model configurations used in pretraining, with 331 million to 1.558 billion trainable parameters.

Model	Layers	Attention heads	Hidden width	Trainable parameters
GPT-2 XL	48	25	1600	1,557,611,200
GPT-2 Large	36	20	1280	774,030,080
OPT-350M	24	16	1024	331,196,416

Single-seed experiments, including the memory measurements in Figure 6 and Table 4, use eight NVIDIA GeForce RTX 5090 GPUs with approximately 32 GiB per device. For the multi-seed experiments, GPT-2 XL uses eight RTX 5090 GPUs, while GPT-2 Large and OPT-350M use eight NVIDIA Quadro RTX 6000 GPUs.

The software environment comprises PyTorch 2.10.0+cu128, Transformers 5.13.0, Datasets 5.0.0, Tokenizers 0.22.2, and NumPy 2.4.1, with NVIDIA driver 595.71.05. The PyTorch build uses CUDA 12.8.

Data and evaluation. We evaluate the optimizers on subsets of three language-modeling datasets: TinyStories, FineMath-4+, and FineWeb-Edu. FineWeb-Edu uses the `sample-10BT` configuration. Each subset contains approximately 100,000 documents after removing exact duplicates, duplicates identified after text normalization, and near-duplicates. Near-duplicate detection compares sets of five consecutive lexical tokens and removes documents with Jaccard similarity of at least 85% to a retained document. A random partition with seed 42 assigns 90% of the retained documents to training and 10% to validation. Before model tokenization, validation documents sharing an identical passage of at least 50 consecutive lexical tokens with a training document are removed. Lexical tokens are defined here as words and punctuation marks.

The remaining documents are encoded using each model's native tokenizer, terminated with an end-of-sequence token, and concatenated separately within each split. A second overlap check operates on the resulting token streams. Candidate validation windows are excluded if they contain an exact match of at least 64 consecutive model tokens with the training stream. Unlike lexical tokens, these model tokens can represent word fragments. From the remaining candidates, 1,000 fixed, nonoverlapping windows are selected for evaluation. Each validation window may span multiple documents and provides 1,024 input tokens with 1,024 corresponding next-token prediction targets. All optimizers use the same training and validation partitions and fixed validation sequences for each dataset and model. Evaluation is performed at initialization and every 50 optimizer updates, with each evaluation covering 1,024,000 next-token prediction targets.

Metrics and aggregation. For N evaluated next-token targets x_i and their corresponding contexts c_i, validation loss and perplexity are defined as

$$\ell_{\mathrm{val}}(W) = -\frac{1}{N}\sum_{i=1}^{N} \log p_W(x_i \mid c_i), \qquad \mathrm{PPL}(W) = \exp\big(\ell_{\mathrm{val}}(W)\big).$$

Natural logarithms are used, so validation loss equals log-perplexity and is expressed in nats per target token. Figure 1 plots single-seed validation loss against cumulative training tokens processed. For validation losses y_1, y_2, y_3 from three training seeds, the mean and sample variance are

$$\bar{y} = \frac{1}{3}\sum_{r=1}^{3} y_r, \qquad s^2 = \frac{1}{2}\sum_{r=1}^{3}(y_r - \bar{y})^2.$$

Table 1 reports these statistics using each seed's last recorded validation loss within the 1,800-second comparison window. Figure 2 plots the mean validation loss against elapsed time, with bands indicating one sample standard deviation above and below the mean. For each optimizer, seed trajectories are linearly interpolated at common time points within the interval covered by all three seeds. The bands represent variation across training seeds.

Training and optimizer settings. All four methods use eight GPU workers, each holding a complete model replica, with a global batch size of 64 sequences. Each worker processes one

sequence per microbatch and accumulates gradients over eight microbatches, giving 65,536 training tokens per optimizer update. Gradients are accumulated locally on each worker and clipped to a total Euclidean norm of at most 1 before the optimizer step. Learning rates remain constant, with no warmup or decay; dropout and weight decay are zero. Parameters, gradients, and optimizer states are stored in FP32. Computation uses BF16 autocast where supported and FP16 autocast with synchronized dynamic loss scaling otherwise. The RTX 5090 runs use BF16 autocast with TF32 enabled. Training uses non-reentrant gradient checkpointing and scaled dot-product attention, with vocabulary projection and loss computation performed in chunks of 256 token positions.

SignMuon uses momentum 0.9 and 10 Newton–Schulz iterations with Frobenius normalization and $\varepsilon = 10^{-8}$. The matrix transform is applied to all trainable parameter matrices, including embedding matrices. Parameter vectors, such as biases and layer-normalization weights, use signed momentum. SignAdam uses bias-corrected first and second moments with coefficients $(0.9, 0.999)$ and $\varepsilon = 10^{-8}$. SignLion uses coefficients 0.9 for the update direction and 0.99 for the momentum update. SignSGD uses no momentum.

Communication. All four methods form directions from local gradients and aggregate their signs through packed AllGather and majority vote. Each worker packs eight signs per byte into uint8 buffers, with at most 1,048,576 parameter elements per communication bucket. Unpacked signs are accumulated in an int16 vote buffer. Zero local directions and tied votes map to $+1$.

Learning-rate selection. For each architecture and optimizer, we searched learning rates in $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ grid. Table 3 lists the selected rates, which remain fixed across the three datasets.

Table 3: Selected pretraining learning rates, fixed across the three datasets.

Model	SignMuon	SignAdam	SignLion	SignSGD
GPT-2 XL	10^{-4}	10^{-5}	10^{-5}	10^{-4}
GPT-2 Large	10^{-4}	10^{-4}	10^{-4}	10^{-4}
OPT-350M	10^{-5}	10^{-5}	10^{-5}	10^{-5}

Budgets and seeds. Comparisons use validation measurements recorded within the first 1,800 seconds of elapsed time, including training, validation, and communication but excluding model and data setup. The number of completed updates varies by optimizer and seed. The single-seed experiments use a fixed random seed of 1337. For GPT-2 XL, these runs are combined with runs using two additional randomly selected seeds. GPT-2 Large and OPT-350M share a set of three randomly selected seeds. Within each architecture, the same seeds are used across all datasets and optimizers, with their values recorded in the experiment configurations in the code repository. The seed for data sampling, document splitting, and validation selection is fixed at 42. Runtime comparisons are restricted to experiments using the same architecture and GPU configuration.

A.7 Additional nanoGPT comparisons

We compare Frobenius and spectral scaling in SignMuon and evaluate distributed weak scaling using nanoGPT (Figures 4 and 5). The model consists of six Transformer layers with six attention heads per layer, embedding dimension 384, context length 512, and approximately 29.94 million parameters excluding positional embeddings. Text is encoded using the GPT-2 tokenizer.

The normalization comparison evaluates five combinations of learning rate and Newton–Schulz iteration count, pairing Frobenius and spectral scaling at each setting. Both methods normalize the momentum matrix before the Newton–Schulz iterations: Frobenius scaling uses its Frobenius norm, while spectral scaling uses an estimate of its largest singular value. The training iteration limit is 1,500, with a linear learning-rate warmup spanning 2,000 iterations.

The weak-scaling experiments use 1, 4, 8, and 16 GPUs while holding the workload per GPU fixed. Each GPU processes 30 sequences of 512 tokens per microbatch with two gradient accumulation steps, giving 30,720 tokens per GPU per update. SignMuon, SignAdam, and SignSGD share a target

learning rate of 10^{-3}, linear warmup, and zero weight decay. SignMuon uses ten Newton–Schulz iterations with spectral scaling.

A.8 Memory measurement details

Figure 6 reports GPT-2 XL memory measurements on TinyStories with seed 1337. For each method we take the largest recorded training-allocation value and persistent-state value. These are per-GPU values, not a sum over eight GPUs. The trainer resets allocation peaks before updates, separates training from validation, and reduces memory statistics with a maximum across workers. Persistent state counts optimizer tensors only; training peaks also include parameters, gradients, activations, and temporary workspaces.

Table 4 extends Figure 6 to OPT-350M and GPT-2 Large. Across all three models, SignMuon and SignLion have the same persistent-state footprint, approximately half that of SignAdam, while SignSGD stores no optimizer state. Peak training allocation includes optimizer state, model parameters, gradients, activations, and temporary tensor allocations. We therefore report optimizer-state size separately from peak training allocation.

Table 4: TinyStories memory measurements (seed 1337; GiB per GPU). SignMuon and SignLion use the same optimizer-state memory, half that of SignAdam; SignSGD has no persistent state.

	OPT-350M		GPT-2 Large		GPT-2 XL	
Method	Peak	State	Peak	State	Peak	State
SignMuon	4.528	1.234	10.526	2.883	21.027	5.803
SignAdam	5.762	2.468	13.504	5.767	26.963	11.605
SignLion	4.528	1.234	10.540	2.883	21.034	5.803
SignSGD	3.294	0.000	7.606	0.000	15.124	0.000

For P FP32 parameters, the stored tensors occupy $4P$ bytes for SignMuon and SignLion, $8P$ bytes for SignAdam, and zero for SignSGD. In GPT-2 XL this gives approximately 5.80, 11.61, and 0 GiB, respectively. Peak allocation is workload and implementation dependent; the persistent-state formula alone does not predict it.

We report allocated GPU tensor memory in GiB (2^{30} bytes). CUDA reserved memory and external device-level telemetry are different metrics and are not substituted for it.

B Five-minute multi-seed nanoGPT experimental setup

We evaluate learning-rate sensitivity and optimization performance on TinyStories, FineMath-4+, and FineWeb-Edu under a five-minute wall-clock budget per run.

B.1 Model, hardware, and experimental design

The experiments comprise 144 runs across three datasets, four signed optimizers, learning rates $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$, and training seeds $\{42, 123, 2026\}$. Each run uses a single NVIDIA Quadro RTX 6000 GPU.

The randomly initialized GPT uses six Transformer layers, six attention heads, embedding dimension 384, context length 512, zero dropout, and no linear or layer-normalization biases. Token embeddings and the language-model head share weights. The GPT-2 tokenizer has 50,257 tokens, and the model vocabulary is padded to 50,304. The model has approximately 30.14 million parameters (29.94 million excluding positional embeddings).

Table 5: Training and optimizer settings for the five-minute nanoGPT experiments.

Setting	Value
Batching	12 sequences per microbatch, five accumulation steps
Tokens per update	$12 \times 5 \times 512 = 30{,}720$ on one GPU
Precision	FP16 autocast with dynamic loss scaling; FP32 model parameters
Compilation	Disabled
Schedule	Constant learning rate; no warmup; weight decay zero
Gradient clipping	Norm threshold 1.0, after unscaling
SignMuon	$\beta = 0.9$; 10 Newton–Schulz iterations in FP32; Frobenius normalization; $\epsilon = 10^{-12}$
SignAdam	$(\beta_1, \beta_2) = (0.9, 0.999)$; $\epsilon = 10^{-8}$
SignLion	$(\beta_1, \beta_2) = (0.9, 0.95)$
SignSGD	No momentum and no persistent optimizer tensor
Logging / validation	Every 10 / 50 updates; final validation at the stopping step
Evaluation sampling	20 batches of 12 sequences per split; fixed sampling generators

The sign operators use $\mathrm{sign}(0) = 0$. Python, NumPy, and CUDA random generators use the specified training seed. TF32 is enabled and cuDNN benchmarking is disabled. Deterministic algorithms are requested where supported, but attention kernels can still introduce nondeterminism.

B.2 Datasets and sampling

Dataset preparation uses fixed revisions and removes exact duplicates after case folding and whitespace normalization. FineMath-4+ and FineWeb-Edu are sampled through a 10,000-document shuffle buffer, retaining at most 100,000 unique nonempty documents each. The TinyStories subset contains 104,919 documents after deduplication. A random document split with seed 42 assigns approximately 10% of each subset to validation. An end-of-text token is appended to each encoded document, and tokens are stored as uint16 memory-mapped streams separately for training and validation.

Table 6: Dataset sizes after deduplication and document-level splitting for the nanoGPT experiments.

Dataset	Train docs	Validation docs	Train tokens	Validation tokens
TinyStories	94,346	10,573	20,741,427	2,310,099
FineMath-4+	89,951	10,049	138,102,256	15,826,128
FineWeb-Edu	89,951	10,049	93,789,022	10,257,169

Training samples random contiguous token windows. Evaluation resets a dedicated CPU sampling generator on every call: seed 42 for training-loss estimation and 43 for validation. It therefore uses matched validation windows across methods, learning rates, and training seeds without advancing the training sampler. Each validation uses $20 \times 12 \times 512 = 122{,}880$ target tokens.

B.3 Timing and result selection

The 300-second budget begins after initial evaluation and includes periodic validation and logging. Training stops at the first update boundary after the budget is reached, so measured durations can slightly exceed 300 seconds. Setup, dataset preparation, and final validation are excluded from the budget.

For each dataset and method, the learning rate is selected by the lowest mean final validation loss across the three seeds. We report the mean and sample standard deviation over those same seeds. Appendix D.3 reports results for all four learning rates, and Appendix D describes the construction of the learning curves.

C Implemented signed optimizer algorithms

The following algorithms specify the zero-weight-decay setting used in both campaigns. At update t, worker r first accumulates its local microbatch gradients, unscales them when using FP16, and clips

the resulting gradient to norm 1. Denote a clipped parameter-block gradient by $G_t^{(r)}$ and the shared parameter by W_t. All workers initialize at the same W_0; persistent momentum and second-moment buffers start at zero and match their parameters. We use zero-based updates, as in Algorithm 1. Powers, square roots, and divisions in the Adam recurrence act entrywise; $A^{\odot 2}$ denotes entrywise squaring.

C.1 Local signs and distributed majority vote

Define $\mathrm{Vote}(D_t^{(1)}, \dots, D_t^{(M)})$ according to the executed campaign. For single-worker nanoGPT, $\mathrm{Vote}(D) = \mathrm{sign}(D)$, including $\mathrm{sign}(0) = 0$. For eight-worker 5090 training, form

$$S_t^{(r)} = 2\,\mathbf{1}\{D_t^{(r)} \geq 0\} - 1, \qquad \overline{S}_t = 2\,\mathbf{1}\left\{\sum_{r=1}^{M} S_t^{(r)} \geq 0\right\} - 1.$$

Each worker packs eight local sign entries per byte, AllGathers the packed buffers, and accumulates unpacked signs into an int16 vote buffer. All workers then apply the same $\overline{S}_t$. The optimizer state remains local; gradients are not averaged before constructing directions. This shared vote operation is used by all four algorithms below.

C.2 SignAdam, SignLion, and SignSGD

Algorithms 6, 7, and 8 use the same clipped local gradients and vote operation, differing in how they form the direction and what state they retain. SignAdam signs its bias-corrected moment ratio; SignLion forms its direction before updating the stored momentum; SignSGD directly signs the current gradient.

Algorithm 6 SignAdam: signed bias-corrected Adam direction

Require: η; $(\beta_1, \beta_2) = (0.9, 0.999)$; $\epsilon = 10^{-8}$

Initialize $M_0^{(r)} \leftarrow 0$, $V_0^{(r)} \leftarrow 0$.
for $t = 0, 1, \dots$ **do**
  Compute the accumulated, unscaled, clipped local gradient $G_t^{(r)}$.
  $M_{t+1}^{(r)} \leftarrow \beta_1 M_t^{(r)} + (1-\beta_1) G_t^{(r)}$.
  $V_{t+1}^{(r)} \leftarrow \beta_2 V_t^{(r)} + (1-\beta_2)(G_t^{(r)})^{\odot 2}$.
  $\widehat{M}_{t+1}^{(r)} \leftarrow M_{t+1}^{(r)}/(1-\beta_1^{t+1})$; $\widehat{V}_{t+1}^{(r)} \leftarrow V_{t+1}^{(r)}/(1-\beta_2^{t+1})$.
  $D_t^{(r)} \leftarrow \widehat{M}_{t+1}^{(r)}/(\sqrt{\widehat{V}_{t+1}^{(r)}} + \epsilon)$, entrywise.
  $\overline{S}_t \leftarrow \mathrm{Vote}(D_t^{(1)}, \dots, D_t^{(M)})$.
  $W_{t+1} \leftarrow W_t - \eta \overline{S}_t$.
end for

SignAdam retains both first and second moments in the executed implementation. For finite entries, its positive denominator and bias-correction factors do not change the sign of the first moment. This algebraic observation does not remove the second-moment tensor from the measured implementation.

Algorithm 7 SignLion: interpolate the old momentum, then update its stored EMA

Require: η; $\beta_1 = 0.9$; $\beta_2 = 0.99$

Initialize $M_0^{(r)} \leftarrow 0$.
for $t = 0, 1, \dots$ **do**
  Compute the accumulated, unscaled, clipped local gradient $G_t^{(r)}$.
  $D_t^{(r)} \leftarrow \beta_1 M_t^{(r)} + (1-\beta_1) G_t^{(r)}$.
  $M_{t+1}^{(r)} \leftarrow \beta_2 M_t^{(r)} + (1-\beta_2) G_t^{(r)}$.
  $\overline{S}_t \leftarrow \mathrm{Vote}(D_t^{(1)}, \dots, D_t^{(M)})$.
  $W_{t+1} \leftarrow W_t - \eta \overline{S}_t$.
end for

Both beta coefficients act on one momentum buffer. The direction uses the old buffer before the β_2 update, so SignLion has one persistent tensor per parameter.

Algorithm 8 SignSGD: signed local gradient with no momentum

Require: Learning rate η; no persistent optimizer tensors
1: **for** $t = 0, 1, \ldots$ **do**
2: Compute the accumulated, unscaled, clipped local gradient $G_t^{(r)}$.
3: $\overline{S}_t \leftarrow \mathrm{Vote}(G_t^{(1)}, \ldots, G_t^{(M)})$.
4: $W_{t+1} \leftarrow W_t - \eta \overline{S}_t$.
5: **end for**

These pseudocodes describe the implementations associated with the current experiments. The separate historical weak-scaling configuration in Appendix A.7 uses the normalization specified for that experiment.

D ADDITIONAL NANOGPT PRETRAINING RESULTS

The experiments follow the five-minute training and evaluation protocol in Appendix B.

D.1 PERFORMANCE UNDER A FIVE-MINUTE BUDGET

Table 7 summarizes final validation performance at the learning rate selected for each dataset and optimizer. Perplexity is computed for each seed before averaging.

Table 7: Final nanoGPT results at the selected learning rate η under a five-minute wall-clock budget. Loss (nats) and perplexity show mean $\pm$ sample standard deviation over three training seeds. Training tokens (millions) and measured run durations (seconds) are means; durations exclude final validation.

Dataset	Method	η	Validation loss	Perplexity	Tokens (M)	Time (s)
TinyStories	SignMuon	10^{-3}	1.999 ± 0.004	7.38 ± 0.03	29.20	300.29
TinyStories	SignAdam	10^{-4}	2.204 ± 0.005	9.06 ± 0.05	36.99	300.16
TinyStories	SignLion	10^{-4}	2.132 ± 0.014	8.43 ± 0.12	36.99	300.16
TinyStories	SignSGD	10^{-4}	3.425 ± 0.016	30.74 ± 0.51	37.42	300.12
FineMath-4+	SignMuon	10^{-3}	3.470 ± 0.012	32.13 ± 0.40	29.19	300.11
FineMath-4+	SignAdam	10^{-4}	3.898 ± 0.007	49.33 ± 0.33	36.88	300.49
FineMath-4+	SignLion	10^{-4}	3.658 ± 0.023	38.81 ± 0.88	37.02	300.17
FineMath-4+	SignSGD	10^{-4}	5.393 ± 0.007	219.97 ± 1.64	37.44	300.13
FineWeb-Edu	SignMuon	10^{-3}	4.696 ± 0.012	109.52 ± 1.26	29.03	300.20
FineWeb-Edu	SignAdam	10^{-4}	5.193 ± 0.014	180.08 ± 2.51	36.86	300.27
FineWeb-Edu	SignLion	10^{-4}	5.050 ± 0.018	156.06 ± 2.87	36.92	300.11
FineWeb-Edu	SignSGD	10^{-4}	6.450 ± 0.003	632.91 ± 1.70	37.42	300.14

SignMuon achieves the lowest mean final validation loss on each dataset, processing about 29 million tokens versus approximately 37 million for the other methods.

D.2 TOKEN EFFICIENCY AND ESTIMATED WALL-CLOCK TIME

Figure 8 compares validation loss against cumulative training tokens and estimated elapsed time. After k updates, each run has processed $N(k) = 30{,}720k$ training tokens. At each selected learning rate, token curves average the losses at evaluation checkpoints shared by all three seeds, ending at their last common 50-update evaluation. Final results at each run's stopping point are reported separately in Table 7.

Elapsed time is measured at the end of each run but is not recorded at every validation checkpoint. The timing accumulator sums durations from updates logged every ten iterations and cannot recover intermediate elapsed times. For seed r, let T_r denote the measured final duration and K_r the number of completed updates. The estimated elapsed time at update k is

$$\widehat{t}_r(k) = T_r \frac{k}{K_r}.$$

This approximation assumes a constant average update rate and distributes validation and logging overhead uniformly over the trajectory. Seed trajectories are linearly interpolated on a five-second grid between 0 and 300 seconds, within their observed estimated-time ranges and without extrapolation. The resulting curves summarize mean progress on this estimated time axis. Bands represent variation across training seeds and exclude uncertainty from time estimation and learning-rate selection.

D.3 Learning-rate sensitivity

Figure 9 compares final validation loss across all four tested learning rates using the full set of 144 runs. The lowest mean loss occurs at 10^{-3} for SignMuon and 10^{-4} for the other methods on all three datasets. Smaller rates yield higher final losses within the five-minute budget, while increasing the rate to 10^{-3} also degrades the other methods' final losses.

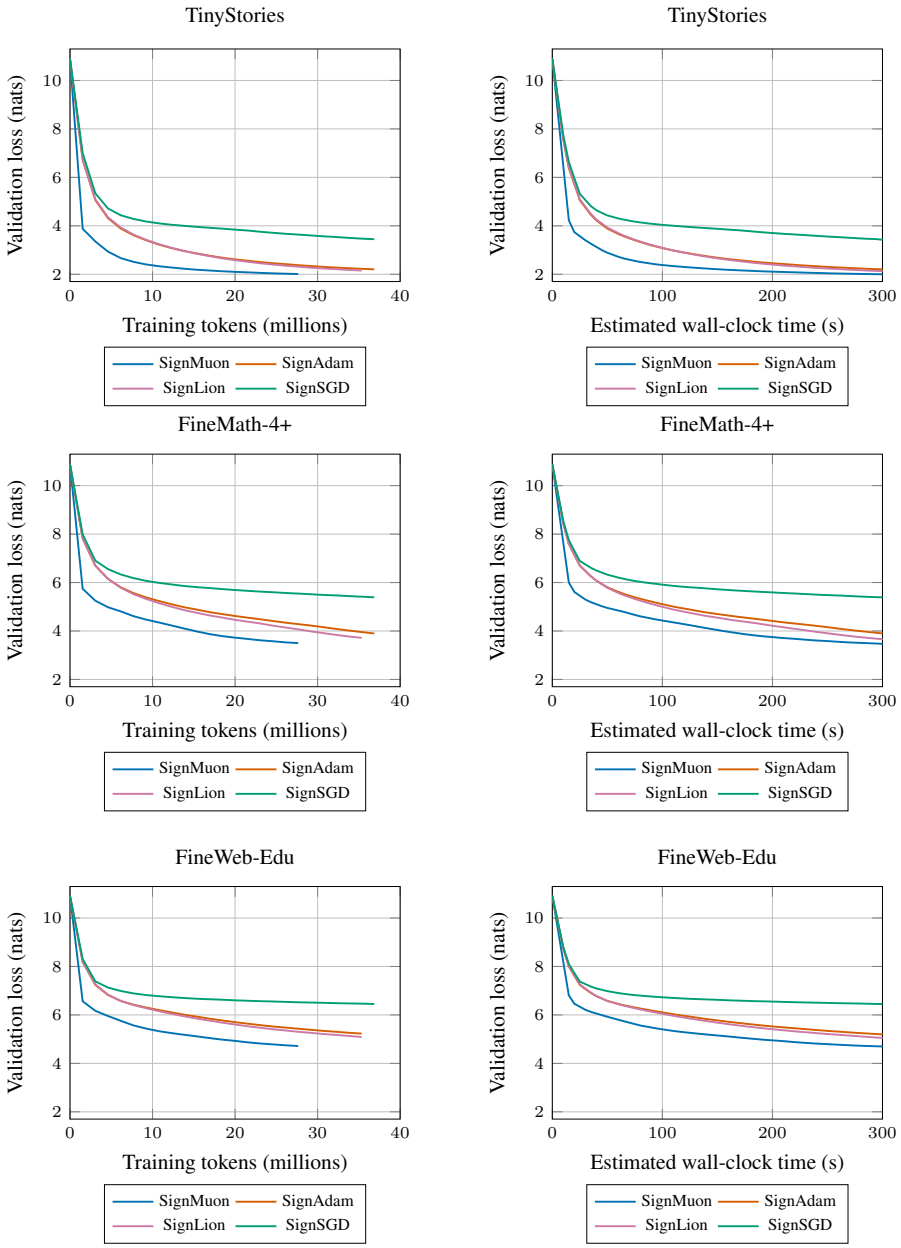

Figure 8: nanoGPT validation loss at the selected learning rates against training tokens (left) and estimated wall-clock time (right). Time coordinates use each run's measured final duration and completed-update count. Curves show means and shading indicates one sample standard deviation over three training seeds. SignMuon has the lowest mean loss at the shared token and estimated-time endpoints. Intermediate time coordinates are estimates, not measured validation timestamps.

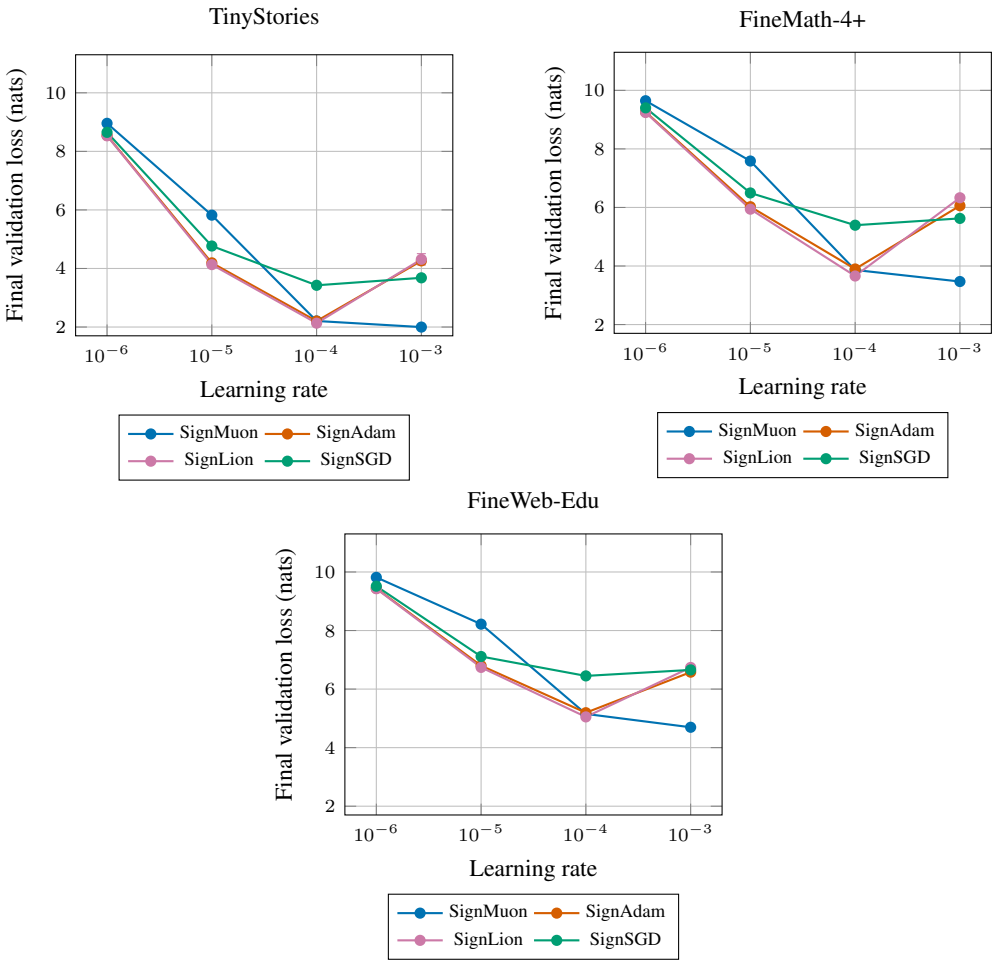

Figure 9: Final nanoGPT validation loss (nats) versus learning rate on a logarithmic horizontal axis. Each point is the mean over three training seeds, with error bars indicating one sample standard deviation. Lines connect the four tested learning rates.

E COMMUNICATION COMPLEXITY

We count the messages and bytes needed to aggregate signs. The model excludes gradient evaluation, local polar computation, packing, and unpacking. Let M denote the worker count and let $d = \sum_{\ell=1}^{L} m_\ell n_\ell$ denote the total number of communicated entries. Concatenation reduces collective calls. The formulas below describe one buffer; for bucketed training, sum the per-bucket costs and byte padding. We exclude scalar checks, logging, and other auxiliary collectives.

E.1 MAJORITY VOTE AND LOCAL OPERATIONS

At iteration t, worker p computes a direction $U_{\ell,t}^{(p)}$ for block ℓ and its binary sign matrix $S_{\ell,t}^{(p)}$. Workers aggregate these signs as

$$\bar{S}_{\ell,t} = \operatorname{sign}\left(\sum_{p=1}^{M} S_{\ell,t}^{(p)}\right). \tag{13}$$

The binary implementation maps a zero sum to $+1$. A different fixed tie rule changes the update at ties but does not change the message size.

Lemma 2 (Majority vote through integer reduction)**.** *Workers can compute* (13) *by encoding the signs as integers, summing them elementwise, and applying a local sign operation. The accumulator must represent every integer in* $[-M, M]$.

Proof. Each coordinate contributes either -1 or $+1$, so its sum lies in $\{-M, -M+2, \ldots, M\}$. Exact integer addition gives this sum, and thresholding it gives the majority vote with the chosen tie rule. □

Once every worker has $\bar{S}_{\ell,t}$, it can normalize the matrix or compute its polar factor locally. For example,

$$D_{\ell,t} = \frac{\bar{S}_{\ell,t}}{\sqrt{m_\ell n_\ell}} \quad \text{or} \quad D_{\ell,t} = \operatorname{polar}(\bar{S}_{\ell,t})$$

requires no further messages before the update $W_{\ell,t+1} = W_{\ell,t} - \eta_t D_{\ell,t}$. The two directions have different optimization properties despite sharing the same communication cost.

E.2 LATENCY–BANDWIDTH MODEL

The cost model separates message startup from byte transfer. This lets us compare the collective paths in Figures 7 and 3 using the same network parameters, and explains why fewer bytes need not give a proportional reduction in total time. Table 8 summarizes the resulting per-worker traffic and round counts.

Definition 3 (Message cost)**.** A point-to-point message of s bytes takes modeled time

$$T_{\alpha,\beta}(s) = \alpha + \beta s,$$

where $\alpha > 0$ is startup latency and $\beta > 0$ is inverse bandwidth in seconds per byte. In this appendix, β denotes the network parameter, not momentum.

We assume simultaneous sending and receiving, no network contention, balanced chunks, and negligible local reduction time. The bandwidth terms count one direction of traffic; sent-plus-received volume is a separate accounting quantity. We ignore padding needed to balance chunks. Within this idealized model, we count a collective's critical-path rounds and per-worker traffic (Rabenseifner, 2004; Thakur et al., 2005; Hoefler et al., 2010). A ring AllReduce performs reduce-scatter followed by AllGather, using $2(M-1)$ rounds. Each phase sends and receives $(1 - 1/M)s$ bytes per worker. Recursive halving/doubling uses $2\log_2 M$ rounds for power-of-two M with the same bandwidth term.

For a pipelined tree, we use the approximations

$$T_{\mathrm{bcast}}(s, M) \approx \alpha\lceil \log_2 M \rceil + \beta s, \tag{14}$$

$$T_{\mathrm{reduce}}(s, M) \approx \alpha\lceil \log_2 M \rceil + \beta s. \tag{15}$$

Without pipelining, sending the complete buffer in every round instead gives a bandwidth term proportional to $\beta s\lceil \log_2 M \rceil$. The tree results below use this idealized pipelined model, omitting additional segment-startup and pipeline-drain costs; they are not exact latency guarantees for a particular backend.

E.3 Integer AllReduce

For a reduction datatype with b bits per entry, the buffer size is

$$s = bd/8 \quad \text{bytes.} \tag{16}$$

Our int8 implementation uses $b = 8$, so $s = s_8 = d$. A packed one-bit representation cannot directly carry the integer partial sums of an ordinary SUM AllReduce. We analyze packed signs through AllGather in Section E.4.

Theorem 4 (AllReduce communication cost)**.** *Suppose workers concatenate their integer sign tensors into one s-byte buffer and use a datatype that avoids overflow. One AllReduce followed by local thresholding has modeled cost*

$$T_{\mathrm{AR}} \approx \alpha R(M) + 2(1 - 1/M)s\beta, \tag{17}$$

where $R(M) = 2(M - 1)$ for a ring and $R(M) = 2\log_2 M$ for recursive halving/doubling with power-of-two M. Each worker sends $2(1 - 1/M)s$ bytes and receives the same amount. The sum of sent and received bytes is $4(1 - 1/M)s$.

Proof. Lemma 2 reduces the computation to an integer sum. Reduce-scatter and AllGather each send $(1-1/M)s$ bytes per worker. Adding the two phases gives the bandwidth term; counting their rounds gives the latency term. Receives match sends, so bidirectional traffic is twice the sent volume. □

If workers instead call AllReduce separately for each of L blocks, the payload sum stays s but the modeled latency becomes $\alpha LR(M)$. Buffer concatenation therefore matters when startup cost is substantial. This observation concerns call organization, not additional quantization.

For int8, Theorem 4 specializes to

$$s_8 = d, \tag{18}$$

$$T_{\mathrm{AR(int8)}} \approx \alpha R(M) + 2(1 - 1/M)d\beta. \tag{19}$$

The accumulator range limits the use of a fixed-width integer type. With int8 signs and int8 accumulation, $M \leq 127$ avoids overflow for every possible sign sum. The experiments use at most 16 workers.

E.4 Packed AllGather

Worker p represents a positive sign by $b_i^{(p)} = 1$ and a negative sign by $b_i^{(p)} = 0$. Packing these bits into a byte array $B_t^{(p)}$ gives

$$s_1 = \lceil d/8 \rceil \quad \text{bytes.} \tag{20}$$

Each byte belongs to $\{0, \ldots, 255\}$; unused bits in the final byte do not represent parameters. After AllGather, every worker has all M buffers. It counts $c_i = \sum_p b_i^{(p)}$ and computes

$$\sum_p S_t^{(p)}[i] = 2c_i - M, \qquad \bar{S}_t[i] = \mathrm{sign}(2c_i - M).$$

The vote requires local computation only.

Traffic and time. Every worker must receive the other workers' $(M-1)s_1$ bytes. A bandwidth-optimal AllGather has

$$V_{\text{send}}^{\text{AG}} = (M-1)s_1, \tag{21}$$

$$V_{\text{recv}}^{\text{AG}} = (M-1)s_1, \tag{22}$$

$$V_{\text{bidir}}^{\text{AG}} = 2(M-1)s_1. \tag{23}$$

Summing over workers gives $M(M-1)s_1$ bytes sent across the system and the same volume received. The modeled time is

$$T_{\text{AG(1bit)}} \approx \alpha R_{\text{AG}}(M) + (M-1)s_1\beta, \tag{24}$$

where a ring uses $R_{\text{AG}}(M) = M-1$, and recursive doubling uses $\log_2 M$ rounds for power-of-two M.

Local cost. AllGather requires storage for Ms_1 bytes and a vote computation on each worker. Unpacking and summing every sign uses $O(Md)$ integer operations. Byte-oriented or bitwise implementations can process the packed representation, but they still read $O(Ms_1)$ bytes. These operations do not change network traffic, yet their cost grows with M.

E.5 AllGather versus AllReduce

Table 8 distinguishes the local payload from total traffic. For $M > 1$, ignoring byte padding, the ratio of the bandwidth terms is

$$\frac{(M-1)s_1}{2(1-1/M)s_8} \approx \frac{(M-1)d/8}{2(1-1/M)d} = \frac{M}{16}. \tag{25}$$

Packed AllGather therefore has a smaller modeled bandwidth term for $M < 16$, a comparable term at 16, and a larger one above 16. A ring AllGather also uses half as many rounds as a ring AllReduce. Different backend schedules can change the latency comparison.

Table 8: Communication costs for one sign buffer. Here $s_1 = \lceil d/8 \rceil$ and $s_8 = d$. Ignoring byte padding, AllGather has a smaller bandwidth term for $1 < M < 16$, but stores all workers' buffers locally.

Quantity	Packed AllGather	Int8 AllReduce
Local payload	s_1	s_8
Bytes sent	$(M-1)s_1$	$2(1-1/M)s_8$
Bytes received	$(M-1)s_1$	$2(1-1/M)s_8$
Ring rounds	$M-1$	$2(M-1)$
Ring bandwidth term	$(M-1)s_1\beta$	$2(1-1/M)s_8\beta$
Result buffer	Ms_1 bytes before local voting	s_8 bytes of sums

Comparison with float32. A float32 buffer contains $4d$ bytes. Int8 uses one quarter as many bytes, and a packed sign representation uses approximately one thirty-second as many. Only the int8-versus-float32 AllReduce comparison holds the collective fixed. For packed AllGather versus float32 AllReduce, the bandwidth-term ratio is approximately $M/64$. A 32-fold representation reduction therefore does not imply a 32-fold collective speedup.

Comparison with signSGD. SignMuon and signSGD have the same aggregation cost when they use the same encoding and collective. SignMuon adds local polar computation. Its benefit must therefore come from the resulting optimization behavior or from a particular balance between local computation and communication, rather than a smaller sign payload than signSGD.

E.6 Parameter-server variants

The preceding collectives distribute communication among workers. Here we consider two alternative aggregation arrangements: a dedicated server with serialized incoming messages, and a tree that

combines partial sums along its edges. Theorems 5 and 6 quantify how these arrangements change the number of communication rounds and the load on the aggregation endpoint while preserving the majority-vote computation.

Theorem 5 (Serialized parameter-server uplink)**.** *Suppose M workers send s-byte sign buffers to one server, which computes the vote and returns an s-byte result. The server is separate from the M workers. If the uplink serializes the messages and the downlink uses a pipelined tree broadcast over $M+1$ participants, the model gives*

$$T_{\mathrm{PS-star}} \approx M(\alpha+\beta s) + \alpha\lceil \log_2(M+1)\rceil + \beta s.$$

The server receives Ms bytes on the uplink.

Proof. The serialized uplink contains M messages, each costing $\alpha+\beta s$. Add the tree-broadcast cost for the result. If the server also sends the result separately to every worker, its downlink sends Ms bytes; the tree model instead distributes forwarding among workers. □

Theorem 6 (Tree reduction and broadcast)**.** *For $M \geq 2$ participants including the root, the pipelined tree model assigns a reduction of an s-byte integer buffer followed by a broadcast the cost*

$$T_{\mathrm{PS-tree}} \approx 2\alpha\lceil \log_2 M\rceil + 2\beta s.$$

Partial sums must fit the reduction datatype.

Proof. Add the modeled reduction and broadcast costs. Intermediate nodes sum the integer entries; the root thresholds the final sums before broadcasting the vote. □

A server can receive packed sign buffers and unpack them before voting, but an ordinary integer reduction tree needs enough bits to store partial sums. The same distinction between sign encoding and sum representation applies as in AllReduce. These server models also show why a direct serialized uplink becomes a bottleneck as worker count grows.

E.7 Implementation implications

Concatenating sign tensors reduces collective startup costs. AllReduce distributes aggregation across workers, while a serialized server collects all incoming traffic at one endpoint. Packed AllGather reduces each worker's input buffer but adds replicated storage and local vote computation. Local polar computation never adds sign-aggregation messages, although it can dominate an iteration on a fast interconnect. Choosing a collective therefore requires measuring local compute and communication together.

F SignMuon: convergence guarantees and solver error

All optimization results in this appendix concern SignMuon: workers sign their local momentum polar approximations and update with the resulting majority vote. We do not substitute gradient signs for this update. The convergence guarantees require the stated vote-alignment assumptions.

F.1 Setup and descent inequality

Let $W \in \mathbb{R}^{m\times n}$, $d = mn$, $r_0 = \min(m,n)$, and $g_t = \nabla f(W_t)$. The inner product is $\langle A, B\rangle = \mathrm{tr}(A^\top B)$. We write $\|\cdot\|_1$ for the entrywise norm and $\|\cdot\|_*$ for the nuclear norm, dual to $\|\cdot\|_{\mathrm{op}}$. For the compact SVD $X = L\Sigma R^\top$, define $\mathrm{polar}(X) = LR^\top$ on its positive singular subspace and $\mathrm{polar}(0) = 0$. This specifies the choice at deficient rank.

Assumption 7 (Objective and smoothness). *The function f is continuously differentiable, $f(W) \geq f^* > -\infty$, and*

$$\|\nabla f(W) - \nabla f(W')\|_* \leq L_* \|W - W'\|_{\mathrm{op}} \tag{26}$$

for a finite $L_* > 0$. The initial W_0 and finite positive step sizes η_t are deterministic. Write $\Delta = f(W_0) - f^*$.

Take $\lambda = 0$ unless f includes the regularization term. For $0 \leq \beta < 1$, worker p uses

$$\begin{aligned} M_{t+1}^{(p)} &= \beta M_t^{(p)} + (1-\beta) G_t^{(p)}, \qquad M_0^{(p)} = 0, \\ U_t^{(p)} &= \mathrm{PolarNS}(M_{t+1}^{(p)}), \qquad S_t^{(p)} = \mathrm{sign}(U_t^{(p)}), \\ \bar{S}_t &= \mathrm{sign}\Big(\sum_p S_t^{(p)}\Big), \qquad D_t = \bar{S}_t/\sqrt{d}, \qquad W_{t+1} = W_t - \eta_t D_t. \end{aligned} \tag{27}$$

The binary convention sends zero to $+1$, including vote ties. The bounds also allow a ternary zero convention if used consistently. Exact polar factors are the idealized version of the same pipeline. Appendix F.5.2 accounts for the implementation's unnormalized signs and multiple blocks. We assume the local routines return finite real matrices and integer aggregation avoids overflow. We take every horizon $T \geq 1$ to be an integer, and every schedule constant η_0 to be positive. No statement here covers the optional polar transformation *after* the vote. The smoothness assumption is a property of the full objective; we do not infer it from a neural-network architecture. In particular, objectives with nonsmooth activations require separate analysis.

Lemma 8 (Descent inequality). *Under Assumption 7,*

$$f(W+H) \leq f(W) + \langle \nabla f(W), H \rangle + \tfrac{L_*}{2}\|H\|_{\mathrm{op}}^2.$$

In particular, SignMuon satisfies $f(W_{t+1}) \leq f(W_t) - \eta_t \langle g_t, D_t \rangle + L_* \eta_t^2/2$.

Proof. Integrate $\langle \nabla f(W + sH) - \nabla f(W), H \rangle$ over $s \in [0,1]$ and use nuclear–spectral duality and (26). Finally, $\|D_t\|_{\mathrm{op}} \leq \|D_t\|_{\mathrm{F}} \leq 1$. □

F.2 Alignment of the actual vote

Condition on W_t and define

$$q_{ij,t} = \Pr(\bar{S}_{t,ij} \neq \mathrm{sign}(g_{t,ij}) \mid W_t), \quad V_t = \frac{\|g_t\|_1}{\sqrt{d}}, \quad e_t = \frac{1}{\sqrt{d}} \sum_{ij} |g_{t,ij}| q_{ij,t}. \tag{28}$$

Coordinates with $g_{t,ij} = 0$ contribute zero under either sign convention. The probability includes momentum history, gradient sampling, the polar routine, and the vote. Unconditional expectations average over all algorithmic randomness.

Lemma 9 (Coordinate alignment). *For the SignMuon vote, $\mathbb{E}[\langle g_t, D_t \rangle \mid W_t] \geq V_t - 2e_t$. Equality holds for binary signs.*

Proof. For $g_{t,ij} \neq 0$, the binary product $g_{t,ij}\bar{S}_{t,ij}$ equals $|g_{t,ij}|$ on agreement and $-|g_{t,ij}|$ otherwise. A zero sign replaces the latter value by zero and only increases the expectation. Sum over entries and divide by $\sqrt{d}$. □

Theorem 10 (Weighted SignMuon bound). *Under Assumption 7, define $A_T = \sum_{t<T} \eta_t$, $B_T = \sum_{t<T} \eta_t^2$, and $\mathcal{G}_T^\eta = A_T^{-1} \sum_{t<T} \eta_t \mathbb{E} V_t$. Then*

$$\mathcal{G}_T^\eta \leq \frac{\Delta}{A_T} + \frac{L_* B_T}{2A_T} + \frac{2}{A_T} \sum_{t<T} \eta_t \mathbb{E} e_t. \tag{29}$$

If $2\mathbb{E}e_t \le \rho\mathbb{E}V_t + \delta_t$ *for fixed* $0 \le \rho < 1$ *and deterministic* $\delta_t \ge 0$*, then*

$$\mathcal{G}_T^\eta \le \frac{1}{1-\rho}\left[\frac{\Delta}{A_T} + \frac{L_* B_T}{2A_T} + \frac{1}{A_T}\sum_{t<T}\eta_t\delta_t\right]. \tag{30}$$

Proof. Take expectations in Lemma 8, apply Lemma 9, and sum. Lower boundedness gives $\sum_{t<T}\eta_t\mathbb{E}V_t \le \Delta + L_* B_T/2 + 2\sum_{t<T}\eta_t\mathbb{E}e_t$. Insert the relative-error assumption and collect the V_t terms for the second bound. Bounded directions place every finite-horizon trajectory in a deterministic compact set. Continuity of f and ∇f therefore makes the expectations in this proof finite. □

If $A_T \to \infty$, $B_T/A_T \to 0$, and $\sum_{t<T}\eta_t\delta_t/A_T \to 0$, the conditional bound gives $\mathcal{G}_T^\eta \to 0$. For example, $\eta_t = \eta_0/\sqrt{t+1}$ and $\delta_t = 0$ give $O(\log(T+1)/\sqrt{T})$. For a fixed horizon, $\eta_t = \eta_0/\sqrt{T}$ gives $O(T^{-1/2})$ when the average δ_t has that order. For an output index R independent of the run with $\Pr(R = t) = \eta_t/A_T$, the bound is exactly on $\mathbb{E}[\|\nabla f(W_R)\|_1/\sqrt{d}]$. A positive residual gives only a bound on this average stationarity measure, not an eventual neighborhood containing every iterate. Neither a residual upper bound nor a stationarity bound establishes a positive lower bound on training error, zero training error, or convergence of the last iterate.

F.3 The local polar solver and the loss from signing

An operator-norm approximation can become inaccurate along nearly null singular directions without losing much inner product with momentum. A linear minimization oracle (LMO) solves $\min_{\|Z\|_{\mathrm{op}}\le 1}\langle X, Z\rangle$, whose solution includes $-\operatorname{polar}(X)$. Following the objective-value criterion of Qian & Richtárik (2026, Assumptions 2.1–2.2), we distinguish these two errors. We derive the following bounds for SignMuon's cubic routine.

Theorem 11 (Cubic Newton–Schulz on the positive singular subspace)**.** *Let* $X \ne 0$ *have rank* r*, let* $\alpha \ge \|X\|_{\mathrm{op}}$*, and set*

$$Y_0 = X/\alpha, \qquad Y_{k+1} = \tfrac{1}{2}Y_k(3I - Y_k^\top Y_k), \qquad Q = \operatorname{polar}(X).$$

Let $\sigma_{\min}^+(X)$ *be the least positive singular value and* $K \ge 0$ *an integer. For* $\rho = 1 - (\sigma_{\min}^+(X)/\alpha)^2 \in [0, 1)$*,*

$$\|Y_K\|_{\mathrm{op}} \le 1, \qquad \|Y_K - Q\|_{\mathrm{op}} \le \rho^{2^K}. \tag{31}$$

The LMO objective gap also satisfies the bound

$$0 \le \epsilon_K(X) := \|X\|_* - \langle X, Y_K\rangle \le r\alpha(2/3)^K. \tag{32}$$

The latter requires no positive lower bound on $\sigma_{\min}^+(X)/\alpha$*. If* $X = 0$*, choose any* $\alpha > 0$*; all iterates, the compact polar factor, and the gap are zero.*

Proof. Write the compact SVD $X = L\Sigma R^\top$. Induction gives $Y_k = L\operatorname{diag}(y_{i,k})R^\top$ with

$$y_{i,0} = \sigma_i/\alpha \in (0, 1], \qquad y_{i,k+1} = y_{i,k}(3 - y_{i,k}^2)/2.$$

Null singular directions remain zero, including for wide or rank-deficient matrices. For $e_{i,k} = 1 - y_{i,k}^2$,

$$e_{i,k+1} = \tfrac{3}{4}e_{i,k}^2 + \tfrac{1}{4}e_{i,k}^3 \le e_{i,k}^2.$$

Thus $0 < y_{i,k} \le 1$, $e_{i,K} \le \rho^{2^K}$, and $1 - y_{i,K} \le e_{i,K}$. This proves (31) without assuming that $I - Y_k^\top Y_k$ contracts on the nullspace.

For the second bound, set $z_{i,k} = y_{i,k}^{-2} - 1 = e_{i,k}/(1 - e_{i,k})$. If $e_{i,k} > 0$, direct algebra gives

$$\frac{z_{i,k+1}}{z_{i,k}} = \frac{e_{i,k}(3 + e_{i,k})}{(2 + e_{i,k})^2} \le \frac{4}{9}.$$

The inequality follows from $(e-1)(5e+16) \le 0$ for $e \in [0,1]$; $e=0$ remains zero. Since $1-y \le \sqrt{y^{-2}-1}$ on $(0,1]$, with $x_i = \sigma_i/\alpha$ we obtain

$$\sigma_i(1-y_{i,K}) \le \alpha x_i (2/3)^K \sqrt{x_i^{-2}-1} = \alpha(2/3)^K \sqrt{1-x_i^2} \le \alpha(2/3)^K.$$

Summing these nonnegative terms proves (32). □

With $\alpha = \|X\|_{\mathrm{F}}$, (32) yields $\epsilon_K(X) \le r(2/3)^K \|X\|_*$. Thus, for fixed rank and sufficiently large K, the relative inner-product error can be uniformly small even when the least positive singular value approaches zero. A scaling floor $\alpha = \max\{\|X\|_{\mathrm{F}}, \varepsilon_{\mathrm{scale}}\}$ instead gives

$$\epsilon_K(X) \le r(2/3)^K (\|X\|_* + \varepsilon_{\mathrm{scale}}).$$

The implementation's additive scaling $\alpha = \|X\|_{\mathrm{F}} + \varepsilon_{\mathrm{scale}}$ obeys the same bound. Both choices introduce an additive tolerance. These are exact-arithmetic statements for the cubic polynomial and a certified upper scaling bound. They do not certify finite power-iteration norm estimates, other Newton–Schulz polynomials, or floating-point rounding.

Proposition 12 (A SignMuon loss decomposition)**.** *For one worker, let M be its current momentum, U satisfy $\|U\|_{\mathrm{op}} \le 1$, and $D = \mathrm{sign}(U)/\sqrt{d}$. In particular, $U = Y_K(M)$ qualifies under the scaling assumptions of Theorem 11. Define*

$$\epsilon_{\mathrm{NS}} = \|M\|_* - \langle M, U\rangle, \qquad \chi_{\mathrm{sign}} = \big[\langle M, U-D\rangle\big]_+, \quad [a]_+ = \max\{a,0\}.$$

Then $\epsilon_{\mathrm{NS}} \ge 0$ and, for any gradient g,

$$\langle g, D\rangle \ge \|g\|_* - 2\|g-M\|_* - \epsilon_{\mathrm{NS}} - \chi_{\mathrm{sign}}. \tag{33}$$

Proof. Feasibility $\|U\|_{\mathrm{op}} \le 1$ and duality imply nonnegativity of the oracle gap. Also,

$$\langle M, D\rangle = \|M\|_* - \epsilon_{\mathrm{NS}} - \langle M, U-D\rangle \ge \|M\|_* - \epsilon_{\mathrm{NS}} - \chi_{\mathrm{sign}}.$$

Use $\langle g-M, D\rangle \ge -\|g-M\|_*$ and $\|M\|_* \ge \|g\|_* - \|g-M\|_*$. □

The feasibility assumption is substantive. For example, $X = \mathrm{diag}(1,4)$ scaled by an underestimate $\alpha = 1$ gives $Y_1 = \mathrm{diag}(1,-26)$, which lies outside the spectral unit ball. A finite power-iteration estimate does not certify Proposition 12. The stationarity bounds based directly on the final signs remain valid under their own alignment conditions, because their direction bound does not require U to be feasible.

The signing loss includes the change of scale from the spectral-ball direction U to $D = \mathrm{sign}(U)/\sqrt{d}$ as well as the change of signs. The solver bound controls ϵ_{NS}, not χ_{sign}. An exact polar factor has zero oracle gap; the separate signing term still enters the alignment condition. If

$$\epsilon_{\mathrm{NS}} + \chi_{\mathrm{sign}} \le (1-\kappa)\|M\|_* + \varepsilon_t, \qquad 0 < \kappa \le 1, \tag{34}$$

then $\langle M, D\rangle \ge \kappa\|M\|_* - \varepsilon_t$. The momentum theorem uses this condition to bound the gradient norm.

For distributed SignMuon with every local $U_t^{(p)}$ feasible, apply (33) separately to the local normalized signs $D_t^{(p)}$. Writing D_t for their normalized vote, define

$$\chi_{\mathrm{vote},t} = \left[\left\langle g_t, \frac{1}{M}\sum_p D_t^{(p)} - D_t \right\rangle\right]_+.$$

Averaging the local inequalities gives the valid aggregate bound

$$\langle g_t, D_t\rangle \ge \|g_t\|_* - \frac{1}{M}\sum_p \left(2\|g_t - M_{t+1}^{(p)}\|_* + \epsilon_{\mathrm{NS},t}^{(p)} + \chi_{\mathrm{sign},t}^{(p)}\right) - \chi_{\mathrm{vote},t}. \tag{35}$$

Thus even local alignment does not by itself establish an aligned majority vote. The aggregate bound therefore includes the vote-loss term.

F.4 Conditional fixed-batch convergence for SignMuon

We bound momentum-tracking error for the pipeline in (27). The proof adapts the inexact-oracle approach of Qian & Richtárik (2026, Theorem 3.6), but places its alignment requirement on the *final SignMuon vote*. No condition on the unsigned polar factor can replace this requirement.

Let $(\mathcal{F}_t)$ be the filtration of histories before sampling each step t, including W_t and all momentum buffers. Suppose

$$\bar{G}_t = \frac{1}{M}\sum_p G_t^{(p)}, \qquad \mathbb{E}[\bar{G}_t \mid \mathcal{F}_t] = g_t, \qquad \mathbb{E}[\|\bar{G}_t - g_t\|_{\mathrm{F}}^2 \mid \mathcal{F}_t] \le \bar{\sigma}^2.$$

For a single worker, this is the usual fixed-mini-batch variance bound. Conditional independence and worker errors centered at $\mathbb{E}[G_t^{(p)} \mid \mathcal{F}_t]$, with conditional second moments at most σ^2 give $\bar{\sigma} \le \sigma/\sqrt{M}$. If worker means differ, their average must still equal g_t. Define the mean momentum $\bar{M}_{t+1} = M^{-1}\sum_p M_{t+1}^{(p)}$.

Theorem 13 (SignMuon under momentum–vote alignment)**.** *Under Assumption 7 and the conditional sampling conditions above, use a constant $\eta > 0$ and $\beta \in [0,1)$ for T steps. Let ε_t be finite deterministic numbers. Assume for every $t < T$ that the actual SignMuon vote satisfies*

$$\mathbb{E}\langle \bar{M}_{t+1}, D_t\rangle \ge \kappa \mathbb{E}\|\bar{M}_{t+1}\|_* - \varepsilon_t, \qquad 0 < \kappa \le 1, \quad \varepsilon_t \ge 0. \tag{36}$$

Then, with $\bar{\varepsilon}_T = T^{-1}\sum_{t<T}\varepsilon_t$,

$$\frac{1}{T}\sum_{t<T}\mathbb{E}\|g_t\|_* \le \frac{1}{\kappa}\Bigg[\frac{\Delta}{\eta T} + \frac{L_*\eta}{2} + \bar{\varepsilon}_T + (1+\kappa)\sqrt{r_0}\left(\frac{\beta\|g_0\|_{\mathrm{F}}}{T(1-\beta)} + \frac{L_*\eta\beta}{1-\beta} + \bar{\sigma}\sqrt{\frac{1-\beta}{1+\beta}}\right)\Bigg]. \tag{37}$$

In particular, $\eta = \eta_0 T^{-3/4}$, $\beta = 1 - T^{-1/2}$, and $\bar{\varepsilon}_T = O(T^{-1/4})$ give $O(T^{-1/4})$ expected average nuclear-gradient norm for fixed dimension, variance, and $\kappa > 0$ independent of T.

Proof. Let $\zeta_t = \bar{G}_t - g_t$ and $\bar{M}_0 = 0$. Expanding the momentum recursion yields

$$\bar{M}_{t+1} - g_t = -\beta^{t+1} g_0 + \sum_{s=1}^{t}\beta^{t-s+1}(g_{s-1} - g_s) + (1-\beta)\sum_{s=0}^{t}\beta^{t-s}\zeta_s. \tag{38}$$

The update bound and smoothness imply $\|g_s - g_{s-1}\|_{\mathrm{F}} \le \|g_s - g_{s-1}\|_* \le L_*\eta$. The martingale differences ζ_s have zero cross terms in the expected squared Frobenius norm: for $u < s$, ζ_u is $\mathcal{F}_s$-measurable and $\mathbb{E}[\zeta_s \mid \mathcal{F}_s] = 0$. Thus

$$\mathbb{E}\left\|(1-\beta)\sum_{s=0}^{t}\beta^{t-s}\zeta_s\right\|_{\mathrm{F}} \le \bar{\sigma}\sqrt{\frac{1-\beta}{1+\beta}}.$$

Triangle inequalities in (38) give

$$\mathbb{E}\|\bar{M}_{t+1} - g_t\|_* \le \sqrt{r_0}\left(\beta^{t+1}\|g_0\|_{\mathrm{F}} + \frac{L_*\eta\beta}{1-\beta} + \bar{\sigma}\sqrt{\frac{1-\beta}{1+\beta}}\right).$$

By duality, $\|D_t\|_{\mathrm{op}} \le 1$, and (36),

$$\mathbb{E}\langle g_t, D_t\rangle \ge \kappa\mathbb{E}\|g_t\|_* - (1+\kappa)\mathbb{E}\|g_t - \bar{M}_{t+1}\|_* - \varepsilon_t.$$

Insert this inequality into Lemma 8, telescope, and use $\sum_{t<T}\beta^{t+1} \le \beta/(1-\beta)$. This proves (37). Substitution of the stated schedules gives the claimed order: the objective term, drift term, and variance term have order $T^{-1/4}$; the initialization term has order at most $T^{-1/2}$. □

This is a horizon-dependent family of runs, not a claim about an infinite run with fixed momentum. The condition concerns $\bar{M}_{t+1}$ and the *voted* signs, although workers compute polar factors from their own momenta. It therefore does not assert that the implementation polarizes the mean momentum. With one worker, (34) suffices; with several workers, the vote requires a separate check. A $1/\sqrt{M}$ variance reduction under independent sampling does not imply a corresponding improvement in κ or the signing loss.

Extension to biased inputs. If the gradients entering the buffer are biased, set

$$b_t = \mathbb{E}[\bar{G}_t \mid \mathcal{F}_t] - g_t, \qquad \zeta_t = \bar{G}_t - \mathbb{E}[\bar{G}_t \mid \mathcal{F}_t], \qquad \bar{b}_T = \frac{1}{T}\sum_{t<T} \mathbb{E}\|b_t\|_* < \infty.$$

Assume the same conditional variance bound for the centered ζ_t. Then (37) holds with the additional term $(1+\kappa)\bar{b}_T/\kappa$ on its right-hand side. To prove this, expansion (38) acquires $R_t = (1-\beta)\sum_{s=0}^{t}\beta^{t-s}b_s$. The remaining noise is still a martingale difference. The geometric weights give

$$\frac{1}{T}\sum_{t<T}\mathbb{E}\|R_t\|_* \le \frac{1}{T}\sum_{s<T}\mathbb{E}\|b_s\|_* \sum_{t=s}^{T-1}(1-\beta)\beta^{t-s} \le \bar{b}_T.$$

Insert this estimate into the same alignment and descent inequalities. Thus the fixed-batch rate additionally requires $\bar{b}_T = O(T^{-1/4})$ when the input is biased. A fixed clipping threshold does not imply this decay.

A setting that ensures the additional condition. For one worker, suppose f depends only on the diagonal entries of a square parameter matrix and every sampled gradient is diagonal. Starting from zero momentum, the buffer and all cubic solver iterates remain diagonal. Positive scaling and the scalar recurrence preserve the sign of every nonzero diagonal entry, so

$$\langle M_{t+1}, D_t\rangle = \frac{\|M_{t+1}\|_*}{\sqrt{d}}.$$

Zero diagonal entries contribute nothing; the binary convention on off-diagonal zeros does not affect this inner product or the objective. Thus (36) holds with $\kappa = 1/\sqrt{d}$ and $\varepsilon_t = 0$ for any finite cubic budget, including zero iterations, under a valid scaling bound. This restricted model shows that the condition is satisfiable. It does not establish alignment for dense neural-network gradients or for a majority vote of different worker momenta.

F.5 Smoothness, multiple blocks, and implementation scope

This subsection connects the single-matrix analysis to the multi-block updates in Algorithms 1 and 4. We first compare smoothness constants and account for block-dependent normalization, then state the joint momentum bound in Theorem 14. We then account for regularization, clipping, and finite polar accuracy.

F.5.1 Comparing spectral and Frobenius smoothness

Let $r_0 = \min\{m, n\}$. Denote by $L_{\mathrm{F}}^{\min}$ and $L_{*}^{\min}$ the least admissible global constants in

$$\|\nabla f(W) - \nabla f(W')\|_{\mathrm{F}} \le L_{\mathrm{F}}\|W - W'\|_{\mathrm{F}}$$

and Assumption 7, respectively. When either is finite, both are finite and

$$L_{\mathrm{F}}^{\min} \le L_{*}^{\min} \le r_0 L_{\mathrm{F}}^{\min}. \tag{39}$$

To prove the first inequality, combine $\|A\|_{\mathrm{F}} \le \|A\|_*$ and $\|B\|_{\mathrm{op}} \le \|B\|_{\mathrm{F}}$ with spectral smoothness. Conversely, Frobenius smoothness gives

$$\|\nabla f(W) - \nabla f(W')\|_* \le \sqrt{r_0}L_{\mathrm{F}}\|W - W'\|_{\mathrm{F}} \le r_0 L_{\mathrm{F}}\|W - W'\|_{\mathrm{op}}.$$

Taking the least constants proves (39). For $f(W) = \|W\|_{\mathrm{F}}^2/2$, the constants are $L_{\mathrm{F}}^{\min} = 1$ and $L_{*}^{\min} = r_0$; a rectangular diagonal matrix with r_0 unit singular values attains the latter ratio.

For a normalized sign direction, $\|S/\sqrt{mn}\|_{\mathrm{F}} \le 1$ as well as $\|S/\sqrt{mn}\|_{\mathrm{op}} \le 1$. Thus every single-matrix sign bound above also holds with a valid L_{F} in place of L_* if one uses the Frobenius descent lemma. Spectral normalization alone does not improve the sign method's curvature term. An unquantized polar direction can instead have squared Frobenius norm as large as r_0, so its Frobenius curvature bound involves $r_0 L_{\mathrm{F}}$. We retain the nuclear–spectral formulation to describe the local polar operation, without claiming an automatic curvature advantage for the signed update.

F.5.2 Simultaneous updates of several parameter blocks

Let $W = (W^{(1)}, \dots, W^{(J)})$ contain all jointly updated blocks, with $d_\ell = m_\ell n_\ell$. A vector is a one-column block. Define the product norm and its dual by

$$\|W\|_\Pi = \max_\ell \|W^{(\ell)}\|_{\mathrm{op}}, \qquad \|G\|_{\Pi,*} = \sum_{\ell=1}^{J} \|G^{(\ell)}\|_*.$$

Their inner product sums the blockwise Frobenius inner products. Assume f is lower bounded and continuously differentiable, and impose the joint condition

$$\|\nabla f(W) - \nabla f(W')\|_{\Pi,*} \le L_\Pi \|W - W'\|_\Pi. \tag{40}$$

This condition controls interactions between blocks; smoothness of each block with the others held fixed does not suffice to ignore those interactions.

Choose fixed $c_\ell > 0$, let $C = \max_\ell c_\ell$, and update every block as

$$W_{t+1}^{(\ell)} = W_t^{(\ell)} - \eta_t D_t^{(\ell)}, \qquad D_t^{(\ell)} = c_\ell S_t^{(\ell)}/\sqrt{d_\ell}, \quad S_t^{(\ell)} \in \{-1, 0, +1\}^{m_\ell \times n_\ell}.$$

Let $g_t^{(\ell)} = \nabla_\ell f(W_t)$ and define $q_{ij,t}^{(\ell)} = \Pr(S_{t,ij}^{(\ell)} \ne \mathrm{sign}(g_{t,ij}^{(\ell)}) \mid W_t)$, conditioning on the full tuple W_t. Define

$$V_t = \sum_\ell \frac{c_\ell \|g_t^{(\ell)}\|_1}{\sqrt{d_\ell}}, \qquad E_t = \sum_\ell \frac{c_\ell}{\sqrt{d_\ell}} \sum_{ij} |g_{t,ij}^{(\ell)}| q_{ij,t}^{(\ell)}.$$

For deterministic W_0 and $\eta_t > 0$, the joint generic bound is

$$\frac{1}{A_T} \sum_{t<T} \eta_t \mathbb{E} V_t \le \frac{f(W_0) - f^*}{A_T} + \frac{L_\Pi C^2 B_T}{2A_T} + \frac{2}{A_T} \sum_{t<T} \eta_t \mathbb{E} E_t. \tag{41}$$

Indeed, product-norm duality and (40) give the same integral descent inequality as Lemma 8, with $\|D_t\|_\Pi \le C$. Summing the coordinate alignment inequalities over blocks gives $\mathbb{E}[\langle \nabla f(W_t), D_t \rangle \mid W_t] \ge V_t - 2E_t$. Taking expectations and telescoping proves (41). Bounded directions again place every finite-horizon trajectory in a deterministic compact set, which justifies the expectations.

Choosing $c_\ell = 1$ gives per-block normalized updates. Choosing $c_\ell = \sqrt{d_\ell}$ gives the implementation's unnormalized signs with one common learning rate; then $C^2 = \max_\ell d_\ell$ and V_t is the sum of the blocks' entrywise gradient norms. Thus a common learning-rate rescaling cannot remove all dimension factors for blocks of unequal shape. Here each $S_t^{(\ell)}$ is the actual SignMuon vote for that block. Its error term remains conditional. This extension does not cover additional blocks updated by a different optimizer unless we include their progress terms separately.

F.5.3 Momentum rate for the full unnormalized method

The preceding joint bound concerns sign errors. We now extend the momentum theorem to all blocks, including biased inputs, so that the learning rate and update scales match the displayed algorithms. Use the actual local momentum–polar–sign vote in each block and the update scales c_ℓ defined above. Let $\bar{G}_t^{(\ell)}$ and $\bar{M}_{t+1}^{(\ell)}$ average the gradients entering the buffers and the updated momenta over workers. With respect to the common pre-sampling history $\mathcal{F}_t$, define

$$b_t^{(\ell)} = \mathbb{E}[\bar{G}_t^{(\ell)} \mid \mathcal{F}_t] - g_t^{(\ell)}, \qquad \zeta_t^{(\ell)} = \bar{G}_t^{(\ell)} - \mathbb{E}[\bar{G}_t^{(\ell)} \mid \mathcal{F}_t].$$

Assume finite conditional means and $\mathbb{E}[\|\zeta_t^{(\ell)}\|_{\mathrm{F}}^2 \mid \mathcal{F}_t] \le \bar{\sigma}_\ell^2$. No independence between blocks is necessary. Put $r_\ell = \min(m_\ell, n_\ell)$ and

$$\begin{aligned} H_0 &= \sum_\ell c_\ell \sqrt{r_\ell}\, \|g_0^{(\ell)}\|_{\mathrm{F}}, \qquad R_c = \sum_\ell c_\ell \sqrt{r_\ell}, \\ \Sigma_c &= \sum_\ell c_\ell \sqrt{r_\ell}\, \bar{\sigma}_\ell, \qquad \bar{b}_T = \frac{1}{T} \sum_{t<T} \sum_\ell c_\ell \mathbb{E}\|b_t^{(\ell)}\|_* < \infty. \end{aligned} \tag{42}$$

Theorem 14 (Joint SignMuon momentum bound)**.** *Assume a continuously differentiable lower-bounded f satisfying* (40)*, deterministic W_0, zero initial momentum, and the sampling conditions above. Use fixed $\eta > 0$, common $\beta \in [0, 1)$, and fixed $c_\ell > 0$ for T steps. Suppose the actual scaled votes satisfy*

$$\sum_\ell \mathbb{E}\langle \bar{M}_{t+1}^{(\ell)}, D_t^{(\ell)} \rangle \ge \kappa \sum_\ell c_\ell \mathbb{E}\|\bar{M}_{t+1}^{(\ell)}\|_* - \varepsilon_t \tag{43}$$

for fixed $0 < \kappa \le 1$ and finite deterministic $\varepsilon_t \ge 0$. Let $\Delta = f(W_0) - f^*$ and $\bar{\varepsilon}_T = T^{-1} \sum_{t<T} \varepsilon_t$. Then

$$\begin{aligned} \frac{1}{T} \sum_{t<T} \sum_\ell c_\ell \mathbb{E}\|g_t^{(\ell)}\|_* \le \frac{1}{\kappa} \Bigg[\frac{\Delta}{\eta T} + \frac{L_\Pi C^2 \eta}{2} + \bar{\varepsilon}_T \\ + (1+\kappa) \Bigg(\frac{\beta H_0}{T(1-\beta)} + \frac{L_\Pi C \eta \beta R_c}{1-\beta} + \Sigma_c \sqrt{\frac{1-\beta}{1+\beta}} + \bar{b}_T \Bigg) \Bigg]. \end{aligned} \tag{44}$$

Choosing $c_\ell = \sqrt{d_\ell}$ recovers the unnormalized sign updates with their common implementation learning rate η. With fixed problem constants, $\eta = \eta_0 T^{-3/4}$, $\beta = 1 - T^{-1/2}$, and $\bar{\varepsilon}_T + \bar{b}_T = O(T^{-1/4})$, the left-hand side is $O(T^{-1/4})$.

Proof. Since $\|D_t\|_\Pi \le C$, joint smoothness gives

$$f(W_{t+1}) \le f(W_t) - \eta \sum_\ell \langle g_t^{(\ell)}, D_t^{(\ell)} \rangle + \tfrac{1}{2} L_\Pi C^2 \eta^2.$$

It also controls each block's gradient drift, including cross-block effects:

$$\|g_s^{(\ell)} - g_{s-1}^{(\ell)}\|_{\mathrm{F}} \le \|\nabla f(W_s) - \nabla f(W_{s-1})\|_{\Pi,*} \le L_\Pi C \eta.$$

Apply expansion (38) to each block, with the additional bias convolution from Appendix F.4. The centered noise has zero cross-time terms within each block; the geometric bias weights sum to at most one after averaging over time. Therefore

$$\frac{1}{T} \sum_{t<T} \sum_\ell c_\ell \mathbb{E}\|g_t^{(\ell)} - \bar{M}_{t+1}^{(\ell)}\|_* \le \frac{\beta H_0}{T(1-\beta)} + \frac{L_\Pi C \eta \beta R_c}{1-\beta} + \Sigma_c \sqrt{\frac{1-\beta}{1+\beta}} + \bar{b}_T.$$

For each block, $\|D_t^{(\ell)}\|_{\mathrm{op}} \le c_\ell$. The triangle inequality and (43) imply

$$\sum_\ell \mathbb{E}\langle g_t^{(\ell)}, D_t^{(\ell)} \rangle \ge \kappa \sum_\ell c_\ell \mathbb{E}\|g_t^{(\ell)}\|_* - (1+\kappa) \sum_\ell c_\ell \mathbb{E}\|g_t^{(\ell)} - \bar{M}_{t+1}^{(\ell)}\|_* - \varepsilon_t.$$

Substitute this inequality into joint descent and telescope. This yields (44); the schedules give the stated order. Bounded directions ensure bounded gradients at each finite horizon, and the assumed noise and bias moments make all momentum expectations used here finite. □

This theorem covers unequal block shapes without treating layers as independent objectives. It still requires the final votes to satisfy (43). It does not infer this condition from local polar feasibility or worker independence. Blocks updated by a different optimizer require additional progress terms.

F.5.4 COUPLED REGULARIZATION AND GRADIENT CLIPPING

The implementation adds λW_t to the gradient before momentum. With $\lambda \geq 0$, this is the gradient contribution of the coupled objective

$$f_\lambda(W) = f(W) + \tfrac{\lambda}{2}\|W\|_{\mathrm{F}}^2, \qquad \nabla f_\lambda(W) = \nabla f(W) + \lambda W.$$

If f satisfies Assumption 7, then f_λ has the valid constant $L_{*,\lambda} = L_* + \lambda r_0$, because

$$\|\nabla f_\lambda(W) - \nabla f_\lambda(W')\|_* \leq (L_* + \lambda r_0)\|W - W'\|_{\mathrm{op}}.$$

For several blocks with coefficients $\lambda_\ell \geq 0$, the corresponding joint constant is $L_{\Pi,\lambda} = L_\Pi + \sum_\ell \lambda_\ell r_\ell$: sum the blockwise inequalities $\lambda_\ell\|W^{(\ell)} - W'^{(\ell)}\|_* \leq \lambda_\ell r_\ell \|W - W'\|_\Pi$. The regularized objective remains lower bounded by f^*. Adding the deterministic term λW_t to an unbiased estimator preserves its conditional variance and centered noise distribution, but changes its mean to $\nabla f_\lambda(W_t)$. A SignMuon alignment condition using this target concerns stationarity of f_λ, not of the unregularized f. This operation differs from decoupled parameter shrinkage. Subsequent momentum and polar transformation still require their own alignment analysis.

Global norm clipping with threshold $\tau > 0$ maps Z to $C_\tau(Z) = Z \min\{1, \tau/\|Z\|_{\mathrm{F}}\}$, with $C_\tau(0) = 0$. It preserves the signs of a sampled gradient but can change its conditional mean and its later momentum direction. If $\mathbb{E}[Z_t \mid \mathcal{F}_t] = g_t$ for one matrix, its clipping bias satisfies

$$\|\mathbb{E}[C_\tau(Z_t) \mid \mathcal{F}_t] - g_t\|_* \leq \sqrt{r_0}\,\mathbb{E}[(\|Z_t\|_{\mathrm{F}} - \tau)_+ \mid \mathcal{F}_t]. \tag{45}$$

Indeed, conditional Jensen and $\|A\|_* \leq \sqrt{r_0}\|A\|_{\mathrm{F}}$ reduce this to $\|C_\tau(Z) - Z\|_{\mathrm{F}} = (\|Z\|_{\mathrm{F}} - \tau)_+$. For global clipping of the concatenated gradient, each block's Frobenius change is at most this same total excess norm, so the corresponding blockwise bound also follows. Deterministic coupled regularization added after clipping does not remove the clipping bias relative to ∇f_λ.

Even a scalar unbiased sample taking values 2 and -1 with equal probability has mean $1/2$ but clipped mean zero at $\tau = 1$. A fixed threshold therefore does not supply the vanishing-bias premise of Appendix F.4 or Theorem 14. The sign-error bound can still describe clipped SignMuon directly, subject to its actual vote-alignment condition. The experimental fixed momentum, constant learning rates, and clipping are not the horizon-dependent settings of the fixed-batch rate theorem.

F.5.5 WHEN POLAR APPROXIMATION PRESERVES SIGNS

Let $Q = \mathrm{polar}(X)$ and let U satisfy $\|U - Q\|_{\mathrm{op}} \leq \varepsilon$. Use the same sign convention for U and Q, including at zero. Since

$$|U_{ij} - Q_{ij}| = |e_i^\top (U - Q) e_j| \leq \varepsilon,$$

the two signs agree whenever $|Q_{ij}| > \varepsilon$. For any gradient matrix g, this yields

$$\left|\left\langle g, \frac{\mathrm{sign}(U) - \mathrm{sign}(Q)}{\sqrt{d}}\right\rangle\right| \leq \frac{2}{\sqrt{d}} \sum_{ij} |g_{ij}|\,\mathbf{1}\{|Q_{ij}| \leq \varepsilon\}. \tag{46}$$

To prove the inequality, discard coordinates whose signs agree and use the bound two on the difference of any two signs. The remaining coordinates all lie in the indicated small-entry set. The same estimate holds for the ternary convention.

The bound combines operator-norm accuracy with entrywise margins: all signs agree if $|Q_{ij}| > \varepsilon$ for every entry. Near-zero entries may change sign. For distributed updates, the vote-loss term in (35) separately accounts for aggregation.

F.6 RELATIONSHIP TO RECENT THEORY

Qian & Richtárik (2026) analyze inexact linear minimization through objective-value error. We use that viewpoint for the cubic solver in Theorem 11, separate signing loss in Proposition 12, and require final-vote alignment in Theorem 13. Their results motivate this analysis but do not establish our alignment premise. Our proofs use ordinary joint smoothness and coupled regularization; their generalized-smoothness and decoupled-weight-decay conclusions do not transfer automatically.