DO ORTHOGONALITY CONSTRAINTS HELP LANGUAGE-MODEL TRAINING BEYOND SPHERE OPTIMIZATION?

Anonymous authors
Paper under double-blind review

ABSTRACT

A growing body of work reports benefits from orthogonality constraints in fine-tuning large language models (LLMs). These results raise an open question: do orthogonality constraints also help LLM pre-training? We conduct a systematic empirical study of decoder-only LLM pre-training with models ranging from 360M to 3.1B parameters, trained on nearly 200B tokens. To explore the potential of orthogonality in pre-training, we examine seventeen variants of orthogonality training: some constrain the weight matrix itself, while others use orthogonal factors to reparameterize the weight. Given that requiring a square weight matrix to be strictly orthogonal results in a loss of half of its parameter degrees of freedom, we also investigate block-wise orthogonality, where every d rows of the weight matrix are required to form a row-orthonormal (Stiefel) matrix, with the block size d serving as a hyperparameter. Among block sizes ranging from 1 to the shorter side of the matrix, no $d > 1$ reliably outperforms $d = 1$, and large blocks are clearly worse. However, when $d = 1$, the orthogonality constraint degenerates into sphere optimization, meaning each row direction is optimized on its own unit sphere, which by itself already improves on tuned Muon optimizers. Our conclusion is that, across the realizations we tried, orthogonality constraints do **not** help language-model pre-training beyond sphere optimization. Since this appears to conflict with several prior results, we re-examine five prior methods in pre-training and fine-tuning that report orthogonality improvements. In each case the reported gain is attributable to numerical precision, training horizon, optimizer realization, or per-row norm control rather than to orthogonality between rows.

1 INTRODUCTION

Before the rise of large language models (LLMs), weight orthogonality was used to keep networks with millions of parameters well conditioned and their feature directions nonredundant (Cissé et al., 2017; Huang et al., 2018). For modern pretrained models, orthogonal fine-tuning methods report gains from learning orthogonal transformations of pretrained weights (Qiu et al., 2023; Liu et al., 2024b; Wu et al., 2026) or from constraining low-rank adapter factors to Stiefel manifolds (Li et al., 2025). In LLM pre-training, orthogonality has so far mostly shaped the training dynamics rather than the weights: the Muon optimizer (Jordan et al., 2024; Liu et al., 2025) orthogonalizes the update to each weight matrix, and POET (Qiu et al., 2025) learns orthogonal transformations of a fixed random weight matrix, preserving its singular values. Research blogs (Bernstein, 2025; Su, 2025) derive optimizers that keep the weights themselves on the Stiefel manifold, without large-scale language-model experiments. Yoshihara (2026) evaluates this approach in GPT-2 models with up to 124M parameters and assigns different orthogonality structures to attention and MLP matrices. Whether keeping the weights themselves orthogonal helps LLM pre-training at scale remains open. We study this question systematically with decoder-only models from 360M to 3.1B parameters trained on nearly 200B tokens.

The most direct form of orthogonality training requires an entire weight matrix $W \in \mathbb{R}^{m\times n}$ with $m \leq n$ to have orthonormal rows, $WW^\top = I_m$. This is a very strong regularization. It imposes $m(m+1)/2$ constraints in total, so a square orthonormal matrix loses about half of its degrees of freedom, i.e., the number of effective parameters is roughly halved. We explored two routes for recovering the lost degrees of freedom.

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.

The first route relaxes the orthogonality constraint on W to an orthogonal reparameterization of W. The simplest variant adds per-row magnitudes: the model learns a row-orthonormal factor Q and a diagonal factor D, and the effective weight is $W = DQ$. This reparameterization can represent every matrix whose rows are mutually orthogonal and nonzero, but it recovers only m degrees of freedom. Even with these magnitudes, full-matrix orthogonality substantially worsens validation loss, whether it is enforced by our projected update or by manifold Muon (Bernstein, 2025) (Figure 1; Appendix K). To go one step further, recall that every full-rank matrix admits a polar decomposition; we therefore replace the diagonal factor D with a symmetric positive-definite (SPD) matrix S, so that $W = SQ$. This reparameterization recovers all mn degrees of freedom: S contributes $m(m+1)/2$ and Q contributes $mn - m(m+1)/2$. A triangular factor ($W = TQ$) and a singular-value form ($W = U\Sigma V^\top$) are equally expressive, but in training they recover much less of the constraint penalty than SPD, and the singular-value form recovers almost none.

The second route relaxes the constraint itself. An $n \times n$ square matrix is partitioned into n/d blocks of size $d \times n$, and only each block is required to have orthonormal rows, so the number of constraints falls to $n(d+1)/2$. The block size d becomes a single knob that ranges from $d = 1$, which constrains only the row norms, to the shorter side of the matrix. Increasing d adds exactly one thing: zero inner products between rows in the same block. The two routes combine: each block is reparameterized with per-row magnitudes, $W_k = D_k Q_k$, or with an SPD factor, $W_k = S_k Q_k$. Our main comparison uses per-row magnitudes at every block size, so increasing d changes only the orthogonality between rows. In the special case $d = 1$, the SPD factor reduces to the per-row magnitude, and the constraint reduces to a sphere constraint on each row with a learnable magnitude.

Combining several ways to train orthonormal matrices with several ways to recover the lost degrees of freedom, we test seventeen variants (Table 9) in pre-training Llama-style LLMs of three sizes, 360M, 1.3B, and 3.1B parameters, with token budgets from $2\times$ to $7.77\times$ the Chinchilla-optimal budget. Across all experiments, no block size $d > 1$ reliably improves on $d = 1$. Small blocks are indistinguishable from $d = 1$, and the penalty grows steadily with block size. SPD factors, which restore mixing between rows of a block, recover much of this penalty at moderate block sizes but give no consistent advantage, and the other realizations trail $d = 1$ as well. At $d = 1$ the orthogonality constraint reduces to sphere optimization: each row direction lives on its own unit sphere with a learned magnitude, and no relation between rows is imposed. We thus arrive, from the orthogonality side, at the direction–magnitude geometry of recent row-norm methods (Lion et al., 2026; Hübler et al., 2026; Hägele et al., 2026), and like them we find that it improves on tuned Muon, as well as on MuonH, which fixes the Frobenius norm of each matrix (Wen et al., 2026a), at every scale we trained. Across the realizations we tried, orthogonality between rows adds nothing beyond sphere optimization. However, the implementation of sphere optimization also matters: weight normalization (Salimans & Kingma, 2016) trained with Muon has the same expressiveness but recovers only about a quarter of the improvement over Muon.

This conclusion appears to conflict with several reports of orthogonality gains, so we re-examine five of them in pre-training and fine-tuning. The released implementations of POET and Pion store the AdamW baseline in bfloat16; with fp32 storage, AdamW improves by 80 to 150 milli-loss, POET's short-run advantage shrinks from 31 to 9 milli-loss and reverses with longer training, and Pion trails AdamW by 95 milli-loss at 1.3B. For module-wise constraints, projected diagonal-Gram training trails a reparameterization that targets the same row geometry, and neither diagonal-Gram nor Stiefel blocks improve on $d = 1$ in the module to which they are assigned. In StelLA, replacing Euclidean adapter factors with sphere-constrained ones gains 2.1 accuracy points, and adding Stiefel constraints on top changes accuracy by -0.09. In PSOFT, replacing the orthogonal latent transformation with independently normalized rows does not reduce accuracy. In each case, the reported gain traces to numerical precision, training horizon, optimizer realization, or per-row norm control rather than to orthogonality between rows.

Our contributions are:

- A block-wise orthogonality family whose block size moves from sphere optimization to the largest Stiefel blocks each matrix allows while holding row magnitudes fixed, so that its effect isolates orthogonality between rows.
- A pre-training study from 360M to 3.1B parameters showing that no block size above one and no tested realization improves on sphere optimization, and that sphere optimization itself improves on tuned Muon.

- Re-examinations of five orthogonality methods in pre-training and fine-tuning that trace each reported gain to a factor other than orthogonality between rows.

2 Related Work

Orthogonality in neural-network training. Orthogonal weights have been pursued through tight-frame, reparameterization, penalty, and manifold methods (Cissé et al., 2017; Huang et al., 2018; Bansal et al., 2018; Li et al., 2020), distinct from spectral normalization (Miyato et al., 2018); classical treatments develop the underlying matrix-manifold algorithms (Edelman et al., 1998; Absil et al., 2008). In the RNN study of Vorontsov et al. (2017), soft orthogonality outperformed exact constraints. We revisit this comparison for language-model pretraining with a norm-matched sphere control.

Geometry in parameter-efficient adaptation. Orthogonality can constrain transformations of pretrained weights, SVD-like or initialized LoRA factors, or manifold-optimized adapters (Qiu et al., 2023; Liu et al., 2024b; Wu et al., 2026; Zhang et al., 2023; Büyükakyüz, 2024; Park et al., 2025; Li et al., 2025; Bogachev et al., 2025; Zhang et al., 2026). These constructions act on different objects. Our adaptation study compares sphere and Stiefel constraints within StelLA's two-factor $USV^\top$ parameterization.

Geometry-aware language-model pretraining. Muon orthogonalizes matrix-valued updates, while manifold Muon asks how to optimize a Stiefel-constrained weight (Jordan et al., 2024; Liu et al., 2025; Wen et al., 2026b; Su, 2025; Bernstein, 2025; Solonko et al., 2026); we ask whether that constraint improves over a sphere control with matched gains. Related methods control Frobenius or operator norms, normalize update directions, or impose module-wise Stiefel and DGram geometry (Wen et al., 2026a; Xie et al., 2026; Gu & Xie, 2026; Yoshihara, 2026; An et al., 2026). POET and Pion use orthogonal transformations but do not require orthogonal effective-weight rows (Qiu et al., 2025; Shi et al., 2026).

Normalization and magnitude–direction separation. Weight normalization separates direction from magnitude (Salimans & Kingma, 2016). Muown is the closest prior parameterization to our $d = 1$ control (Lion et al., 2026; Hübler et al., 2026); related work studies row or column gains, broader parameter normalization, and magnitude–direction decompositions in adaptation (Hägele et al., 2026; Velikanov et al., 2026; Loshchilov et al., 2025; Liu et al., 2024a; Hu et al., 2022; Kang et al., 2026). Our $d = 1$ control confirms this norm-and-scale line; the new comparison begins when d increases while per-row gains remain fixed.

Appendix A gives an extended discussion.

3 Method

We build every constrained variant on Muon. Section 3.1 describes a general recipe for training a matrix with orthonormal rows, and Section 3.2 applies it to row blocks of Transformer weights and describes how we return the degrees of freedom that the constraint removes. Appendix D describes the other optimizers and controls we compare.

3.1 Optimizing row-orthonormal matrices with OrthMuon

OrthMuon separates the two jobs of a constrained optimizer: Muon chooses the update direction, and the geometry of the constraint set keeps the parameter feasible. For $m \le n$, the constraint set is the row-Stiefel manifold

$$\mathrm{St}(m, n) = \left\{ Q \in \mathbb{R}^{m \times n} : QQ^\top = I_m \right\}, \tag{1}$$

which is the orthogonal group when $m = n$.

At each step, OrthMuon forms Muon's direction Z_t from the Euclidean gradient, by Nesterov momentum followed by Newton–Schulz orthogonalization (Jordan et al., 2024; Liu et al., 2025). It then removes the part of Z_t that would violate the constraint to first order, by projecting onto the tangent space $T_Q = \{\Delta : \Delta Q^\top + Q\Delta^\top = 0\}$,

$$\Pi_Q(Z) = Z - \mathrm{sym}(ZQ^\top)Q, \qquad \mathrm{sym}(X) = \tfrac{1}{2}(X + X^\top), \tag{2}$$

Input: $Q \in \mathrm{St}(m,n)$, momentum B, gradient G, learning rate η, scale s.

1: $B \leftarrow \mu B + G,\ Z \leftarrow \mathrm{NS}_5(G + \mu B)$	*Muon direction*
2: $A \leftarrow Q - \eta s\,\big(Z - \mathrm{sym}(ZQ^\top)Q\big)$	*tangent step*
3: $Q \leftarrow p(AA^\top - I_m)\,A$	*polynomial retraction*

Algorithm 1: One OrthMuon step for $Q \in \mathrm{St}(m,n)$. Appendix C gives the blockwise version and implementation details.

and takes the step $A_t = Q_t - \eta_t s\,\Pi_{Q_t}(Z_t)$, where s is Muon's shape-dependent scale.

Because the step is tangent, $A_t A_t^\top = I_m + \eta_t^2 s^2\,\Pi_{Q_t}(Z_t)\Pi_{Q_t}(Z_t)^\top$: the iterate leaves the manifold only at second order. We therefore return to the manifold with a polynomial instead of a matrix factorization. The nearest point of $\mathrm{St}(m,n)$ to A_t is its polar factor $(I_m + E_t)^{-1/2}A_t$ with $E_t = A_tA_t^\top - I_m$, and we replace the inverse square root by its fourth-order Taylor polynomial,

$$Q_{t+1} = p(E_t)\,A_t, \qquad p(E) = I_m - \tfrac{1}{2}E + \tfrac{3}{8}E^2 - \tfrac{5}{16}E^3 + \tfrac{35}{128}E^4. \tag{3}$$

The retraction uses only matrix products, the primitive that Newton–Schulz already relies on, with no QR, SVD, or eigendecomposition in the inner loop. Proposition 1 bounds its error and cost (proof in Appendix C).

Proposition 1 (Polynomial retraction) *Let $Q \in \mathrm{St}(m,n)$, let Δ be tangent at Q with $\|\Delta\|_2 \le 1$, and let p_r, $r \ge 1$, be the degree-r Taylor polynomial of $(1+x)^{-1/2}$. Then $E = (Q+\Delta)(Q+\Delta)^\top - I_m = \Delta\Delta^\top$, and $Q^+ = p_r(E)(Q+\Delta)$ satisfies*

$$\big\|Q^+Q^{+\top} - I_m\big\|_2 \le c_r\|\Delta\|_2^{2r+2}, \qquad c_r = \binom{2r+2}{r+1}\Big/2^{2r+1},$$

with equality to leading order as $\Delta \to 0$. In Horner form, Q^+ costs $4m^2n + 2(r-1)m^3$ matrix-multiply FLOPs.

With $\Delta = -\eta_t s\,\Pi_{Q_t}(Z_t)$, the order $r = 4$ used here gives $c_4 = 63/128$ and an error of order η_t^{10}. Raising the order from two to four adds only two $m \times m$ products. We therefore fix $r = 4$ rather than choose the order from $\|E_t\|_F$, which would force a GPU-to-CPU synchronization at every step. A few exact polar corrections over training remove accumulated rounding error. At 1.3B on 8 H100 GPUs, $d = 1$ trains at 97% of Muon's throughput (Appendix C); we hope this implementation serves as a practical reference for orthogonality training at scale.

3.2 Recovering degrees of freedom

Let $W \in \mathbb{R}^{m\times n}$ be a matrix-shaped Transformer weight. We partition its rows into m/d contiguous blocks and keep the directional factor of each block, Q_k, on $\mathrm{St}(d,n)$. The Muon direction and scale are computed for the whole matrix; each block is then projected and retracted independently as in Section 3.1. When $m \le n$, $d = m$ constrains the whole matrix.

Reparameterization. The first route forms each effective block from its orthonormal factor and a magnitude factor. With per-row magnitudes,

$$W_k = D_kQ_k, \qquad D_k = \mathrm{diag}(a_k), \qquad W_kW_k^\top = D_k^2, \tag{4}$$

so the effective rows stay orthogonal while each keeps its own learned norm. To allow mixing between rows, we replace D_k with an SPD factor,

$$W_k = S_kQ_k, \qquad S_k = L_kL_k^\top, \qquad L_k = \mathrm{diag}(a_k) + \mathrm{tril}(M_k,-1). \tag{5}$$

Now $W_kW_k^\top = S_k^2$ is generally dense, so the effective rows are no longer orthogonal, and by the polar decomposition this form can represent any full-row-rank block. We initialize $S_k = I_d$. The magnitude factors a_k and L_k are trained with AdamW. Other left factors, including triangular ($W_k = T_kQ_k$), singular-value ($W_k = U_k\Sigma_kQ_k$ with U_k orthogonal), and other symmetric forms, are equally expressive and are defined in Appendix D.

Block size. The second route shrinks the set of constrained row pairs. With per-row magnitudes at every d, an m-row matrix has $m(d-1)/2$ row pairs constrained to be orthogonal, a fraction $(d-1)/(m-1)$ of all pairs, while the number of row norms and the way they are optimized stay the same. Increasing d therefore changes only the orthogonality between rows. At $d = 1$, the directions form a product of unit spheres (the oblique manifold (Absil & Gallivan, 2006)), the SPD factor reduces to the per-row magnitude, and the model has the same function class as an unconstrained row with no constraint between rows. We call this case sphere optimization.

4 Controlled Pretraining Results

4.1 Experimental design and tuning

We train decoder-only Transformers with 360M, 1.3B, and 3.135B parameters at context length 4096. Muon-family runs scale decoupled weight decay by the Muon learning rate (Liu et al., 2025) and use cosine decay to 0.01 times the peak learning rate. Tuned Muon, which shapes the update spectrum without constraining the trained weight to be orthogonal, serves as our principal unconstrained baseline. Our primary corpus is C4 (Raffel et al., 2020). Budgets are reported as multiples of 20 tokens per parameter, the representative Chinchilla ratio (Hoffmann et al., 2022). Unless stated otherwise, we report final target-step cross-entropy on C4-val, a fixed evaluation set drawn from the C4 validation split. Muon, MuonH, and the $d = 1$ control receive separate main-learning-rate sweeps; unconstrained baselines also tune weight decay. The AdamW-side learning rate is fixed at 0.0012 across the Muon-family arms. For the 360M d-ladder, a bracket over $\{0.001, 0.002, 0.004\}$ selects the common interior value 0.002 for each d, including $d = 32$. Main-text entries are point estimates, averaged over three different seeds. Appendices B–O give additional grids. We report loss differences in milli-loss (10^{-3} nats).

4.2 Block orthogonality provides no measured gain

The headline comparison holds norm treatment and gain granularity fixed. Every $d = 1, 8, 32$ model retains one learned gain per row; increasing d makes rows within each block orthogonal to each other. At both budgets, $d = 8$ changes mean loss by only 0.41–0.53 milli-loss, whereas $d = 32$ increases mean loss by 4.67–5.45 milli-loss (Table 1).

4.3 Expressivity and optimizer controls for $d > 1$

We tested diagonal and matrix-valued gains together with projected diagonal-Gram (DGram) training, which keeps $WW^\top$ diagonal by projection while leaving row norms free. SPD uses $S = LL^\top$, XXT uses $S = XX^\top$, and XXT+diagonal uses $S = \mathrm{diag}(a) + XX^\top$. At 360M, SPD and XXT reduce the deficit from 7.6 milli-loss under one scalar gain per block to 1.0 and 1.4 milli-loss (Table 2), consistent with restoring within-block degrees of freedom, including row mixing. Trained $d = 1$ rows are about as decorrelated as random directions: within 32-row windows, their mean absolute cosine is 0.030 (0.046 in query and key matrices), against 0.025 for random directions in $\mathbb{R}^{1024}$ and 0.033 for tuned Muon and MuonH. Per-row $d = 32$ sets the within-block value to zero, and the SPD gain raises it to 0.052 (Appendix L). Expressivity alone does not decide this: equally expressive triangular and singular-value factors recover much less (Table 25). Larger blocks widen the gap (Figure 1): at $d \geq 128$, every tested per-row and SPD configuration trails $d = 1$ at both 360M and 1.3B, by more at each larger d.

Matrix-valued gains, which allow nonorthogonal effective rows, show no consistent advantage over $d = 1$. At 1.3B, where a five-point main-learning-rate bracket selects 0.001, $d = 32$ SPD is 0.4 milli-loss below $d = 1$; the 3.135B diagnostic is 0.1 milli-loss below $d = 1$. At 360M the sign is not stable: SPD is 1.0 milli-loss worse than $d = 1$ in Table 2 and 3.3 milli-loss better in a separate comparison (Appendix K). In FPO-mix checks, the two are tied at both $2\times$ and $4\times$, within 0.3 milli-loss. At 1.3B, $d = 64$ SPD already trails $d = 1$ by 0.5 milli-loss (Appendix K).

Manifold Muon (Bernstein, 2025; Su, 2025) solves the constrained spectral-norm steepest-descent problem directly but trails $d = 1$ at every tested block size, budget, and scale: by 23–140 milli-loss at 360M and by 33–62 milli-loss at 1.3B (Appendix K). It improves on projected OrthMuon only at the

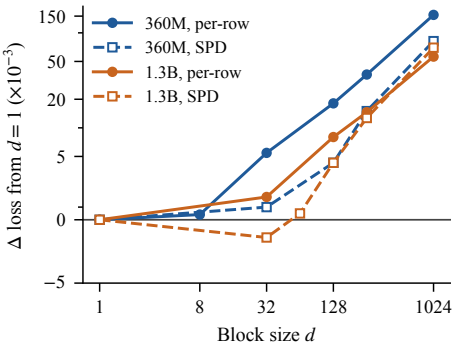
Table 1: Final C4 validation loss (lower is better): the $d = 1$ control beats Muon and MuonH, while $d = 8, 32$ do not improve on it in the matched per-row-gain ladder.

	Trained on C4				Trained on FPO-mix	
Method	360M, 2×	360M, 7.69×	1.3B, 2×	3.135B, 2×	360M, 2×	1.3B, 2×
Muon	2.475	2.372	2.198	2.037	2.682	2.413
MuonH (fixed matrix norm)	2.469	2.363	2.191	2.031	2.677	2.408
OrthMuon $d = 1$ (sphere + gain)	**2.462**	**2.357**	**2.185**	**2.025**	**2.670**	**2.401**
OrthMuon $d = 8$	2.463	**2.357**	2.186	–	–	–
OrthMuon $d = 32$	2.468	2.361	2.186	**2.025**	2.676	2.403

Table 2: Change in final C4 loss from $d = 1$ at $d = 32$, 2× ($\times 10^{-3}$; negative is better; matched $d = 1$ references): full-matrix gains nearly close the 360M gap, but no realization beats $d = 1$ consistently.

Realization at $d = 32$	360M	1.3B	3.135B
Per-row gain	+5.4	+1.8	+0.3
Per-block gain	+7.6	–	–
SPD gain $LL^\top$	+1.0	−0.4	−0.1
XXT gain	+1.4	+0.6	+0.5
Projected DGram	+19.1	–	–
Manifold Muon	+30.5	+42.7	–

Figure 1: The C4 loss penalty grows with block size under per-row and SPD gains (2×; symmetric-log axis; each point uses its matched $d = 1$ reference).



largest tested block size at 360M and trails it at every tested block size at 1.3B, so the negative result is not specific to the projected update.

4.4 The $d = 1$ control is strong but optimizer-dependent

The $d = 1$ control improves over tuned Muon, confirming the positive norm-and-scale results reported by recent work (Lion et al., 2026; Hübler et al., 2026; Hägele et al., 2026). Across three budgets, it lowers validation loss by 13.5–16.9 milli-loss; the unit-row constraint without gain and the learned row magnitudes each contribute about half (Table 3). MuonH, which controls one Frobenius norm per matrix, closely tracks the no-gain $d = 1$ model. Its deficit to $d = 1$ lies mainly in attention: 8.1 milli-loss there versus 2.6 in MLP matrices (Appendix N).

The function class alone does not explain this gain. Applying Muon to an unconstrained latent weight and normalizing rows only in the forward pass reaches 2.4715. It recovers 3.7 of the 13.5 milli-loss improvement over Muon and remains 9.8 milli-loss behind explicit sphere-coordinate optimization. The coordinate system and update dynamics therefore matter alongside norm and scale. Alternative gain coordinates, direct row normalization, and tangent or transported momentum remain within 2.4 milli-loss of the explicit $d = 1$ result (Appendix O).

4.5 Scope across scale and training distributions

Changing the training mixture preserves the matched diagonal-gain ordering. With the FPO-mix training mixture—92.57% DCLM, 6.09% StarCoder, and 1.34% ProofPile (Wen et al., 2026b)—and C4-val evaluation, $d = 1$ again leads MuonH and Muon at 360M and 1.3B, and per-row-gain $d = 32$ at 360M (Table 1; Appendix I). Across six alternate-corpus evaluation sets, C4-trained 360M checkpoints show no consistent ordering among block sizes or baselines. At 2×, $d = 32$ has the lowest mean loss on the FPO mixture, DCLM, and Wikipedia, whereas Muon is lowest on StarCoder, ProofPile, and OLM-arXiv. At 7.69×, $d = 1$ is lowest on the FPO mixture and DCLM, $d = 8$ on StarCoder, ProofPile, and OLM-arXiv, and $d = 32$ on Wikipedia. Relative to Muon, $d = 1$ is within 7 milli-loss on the three natural-language-heavy sets at 2×; at 7.69× it is 39 and 28 milli-loss lower on the FPO mixture and DCLM but 17 milli-loss higher on Wikipedia (Appendix J).

Table 3: Improvement over tuned Muon at 360M ($\times 10^{-3}$; higher is better). A unit-row directional constraint and learned row gains each supply about half of the $d = 1$ gain; normalizing rows only in the forward pass recovers little.

Method	2×	4×	7.69×
Forward-normalized Muon	3.7	–	–
MuonH (fixed matrix norm)	5.8	8.7	9.2
$d = 1$, no gain	6.8	9.1	7.0
$d = 1$, row gain (total)	**13.5**	**16.9**	**15.1**

The strength of the $d = 1$ control also extends beyond 360M. At 1.3B, it improves over Muon by 0.012–0.013 at 2× and 4×, and the diagnostic ordering holds at 1.3B 7.77× (2.107 versus 2.120) and 3.135B 2× (2.025 versus 2.037); Appendix H reports the full scaling table. The per-row $d = 32$ penalty shrinks with scale (Table 2).

5 Complementary Attribution Studies

We next revisit orthogonality-associated methods in pretraining and adaptation. The cases address different attribution questions: POET and Pion test numerical precision and training horizon, DGram tests optimizer realization for a shared constraint set, StelLA supplies a second matched sphere-to-Stiefel comparison, and PSOFT tests the parameterization of a latent transformation. Only the pretraining d-ladder and the StelLA comparison are direct sphere-to-Stiefel comparisons; the other cases address different attribution questions.

Table 4 summarizes the four studies.

5.1 Pretraining studies

POET and Pion: precision and training horizon. POET learns orthogonal transformations of a fixed random base matrix and preserves its initialized singular-value spectrum (Qiu et al., 2025). Pion turns this construction into an optimizer that updates each weight through left and right orthogonal transformations (Shi et al., 2026). Both released implementations store the AdamW baseline's parameters and optimizer moments in bfloat16, and in both cases fp32 storage makes AdamW far stronger (Table 5). In POET's harness at 197M tokens, with independently swept learning rates, bfloat16 raises AdamW loss by 149 milli-loss and POET loss by 127, because POET's factors start at zero or identity and its moments are periodically reset. POET's advantage shrinks from 31.2 to 8.7 milli-loss, and it does not survive longer training: in a separate 130M implementation, POET leads tuned AdamW by 370 and 61 milli-loss at 0.59B and 2B tokens but trails by 8 and 19 milli-loss at 6B and 30.1B. Pion reports a 35-milli-loss advantage over bfloat16 AdamW for a 1.3B model trained on 54B C4 tokens at context length 256, while keeping its own Lie-algebra momentum in fp32. At context length 4096, with the paper's update settings and a Pion learning-rate sweep, Pion trails bfloat16 AdamW by 15.0 milli-loss and fp32 AdamW by 95.4 (Appendix Q). In both cases, storing the baseline in bfloat16 costs AdamW more than the reported advantage of the orthogonal method.

Different Layers, Different Manifolds: constraint versus optimizer. Yoshihara (2026) reports that assigning Stiefel constraints to attention and DGram constraints to MLP matrices lowers GPT-2 validation loss from 3.3855 without constraints and 3.3679 under all-Stiefel training to 3.3544; their Stiefel arm follows the manifold-Muon construction of Bernstein (2025), and the result combines a module assignment with a projected-gradient optimizer. Our controlled comparison fixes a blockwise $d = 32$ DGram geometry and varies its optimizer realization. Projected DGram keeps $WW^\top$ diagonal, whereas $W = \mathrm{diag}(a)Q$ targets the same row geometry but assigns momentum, adaptive scaling, and regularization to different variables. Independent LR–WD tuning lowers projected DGram from 2.545 to 2.481; it remains behind tuned Muon (2.475) and $d = 1$ (2.462), and behind the per-row $d = 32$ reparameterization of the matched ladder (2.468). To test the module assignment, we keep one module type under $d = 1$ and replace the other (Appendix N). Relative to training every matrix with $d = 1$, DGram raises loss by 13.4 milli-loss in MLP matrices and 11.0 in attention matrices, and per-row $d = 32$ raises it by 3.1 and 4.4. Neither constraint improves on $d = 1$ in the module to which Yoshihara (2026) assign it, and the $d = 32$ penalty arises in both module types.

Table 4: Controlled comparisons separate orthogonality from precision, training horizon, optimizer realization, and norm control.

Study	Comparison	Finding
POET, Pion	fp32 AdamW state; longer training	POET: 31 to 9 milli-loss with fp32 state; behind AdamW at 6B and 30.1B tokens. Pion: behind bfloat16 and fp32 AdamW at 1.3B.
Different Layers	DGram projection versus reparameterization; module assignment	Projection trails Muon, $d = 1$, and the $d = 32$ reparameterization. Neither DGram in MLP nor $d = 32$ in attention improves on $d = 1$.
StelLA	Sphere-constrained factor columns	Sphere gains 2.1 points over Euclidean; Stiefel then changes accuracy by -0.09.
PSOFT	Row-sphere transform in the same selected subspace	Row-sphere scores 0.5 points above the implemented transform at validation-selected rates; the ordering reverses with learning rate.

Table 5: fp32 storage makes AdamW much stronger and shrinks or reverses each method's advantage. Entries are final losses; Δ is method minus AdamW ($\times 10^{-3}$; negative favors the method). POET uses the same storage as AdamW; Pion runs once at its released precision.

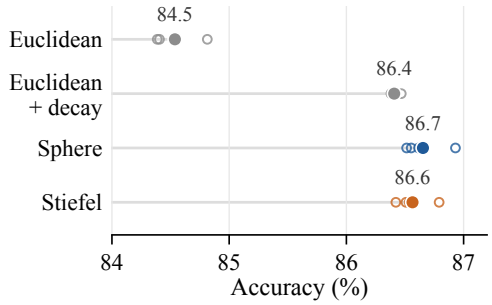
(a) POET, 130M, 0.2B tokens

Storage	AdamW	POET	Δ
bfloat16	3.943	3.912	-31.2
fp32	3.793	3.785	-8.7

(b) Pion, 1.3B, 51B tokens, context 4096

AdamW storage	AdamW	Pion	Δ
bfloat16	2.346	2.361	$+15.0$
fp32	2.265	2.361	$+95.4$

Figure 2: StelLA accuracy: Sphere reproduces the gain over Euclidean factors, and Stiefel adds nothing (filled markers are means, hollow markers individual seeds).



5.2 Adaptation studies

StelLA: norm control. StelLA constrains both low-rank factors in Equation 8 to Stiefel manifolds (Li et al., 2025). In Llama-3-8B commonsense adaptation (Llama Team, 2024), we insert a product-of-spheres control between Euclidean and Stiefel factors. All runs use StelLA's default $S_0 = I$. Sphere improves over Euclidean by 2.12 accuracy points, whereas Stiefel changes Sphere by -0.09 points (Figure 2). All arms use StelLA's default learning rate 5×10^{-4}. In a check over $\{10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$, this is the best tested value for Sphere and Stiefel, while the best tested Euclidean rate, 10^{-3}, lies at the grid edge and remains 1.87 points below Sphere (Appendix R).

The Sphere gain is large, while adding row orthogonality does not improve accuracy. Factor-wise decay (U/V weight decay 1.0, the largest tested value) comes within 0.25 points of Sphere and recovers about 88% of the Euclidean-to-Sphere gain. In an update-rule ablation, a direct-polar Stiefel update improves over the canonical update by about 0.13 points (Appendix R).

PSOFT: orthogonality within principal-subspace adaptation. PSOFT learns an orthogonal transformation restricted to a principal subspace of the pretrained weight (Wu et al., 2026). Our controlled reconstruction holds fixed the selected subspace, rank, target modules, effective initialization, learned magnitude vectors, and training protocol. It compares PSOFT's implemented Cayley/Neumann latent transformation with a dense latent transformation whose rows are optimized independently on spheres. Because the coordinate system and optimizer realization also change, this comparison tests the implemented orthogonal parameterization as a whole. At learning rates selected by validation loss, the row-sphere transform reaches 81.07 average accuracy, compared with 80.55 for PSOFT's implementation and 80.95 for an unconstrained dense transform (Appendix S). Removing orthogonality between rows does not reduce accuracy, and the sphere constraint adds about 0.1 points over the dense transform at the selected rates and none at 8×10^{-4}. The ordering of PSOFT and row-sphere

depends on the learning rate: PSOFT leads by 0.19 points at 4×10^{-4} and trails by 0.64 at 8×10^{-4}. The implemented five-term Neumann transform is also not exactly orthogonal; its singular values range from 0.86 to 1.00.

6 Discussion and Limitations

Stronger block constraints and row-mixing freedom. A monotone conditioning or diversity account predicts that decorrelating more row pairs should help, or at least not hurt. Instead, $d = 8$ has no resolved effect, $d = 32$ is worse in mean loss at both 360M budgets, and, at 360M, matrix gains that restore within-block degrees of freedom, including row mixing, reduce the $d = 32$ penalty from 7.6 to 1.0–1.4 milli-loss (Section 4.3). In trained 360M checkpoints, per-row $d = 32$ and tuned Muon both give 32-row blocks a higher median stable rank than $d = 1$ (25.3 and 20.4 versus 19.2) and a lower median condition number (1.31 and 1.62 versus 1.74), yet both have higher loss (Appendix L).

Optimizer realization matters alongside the constraint. The $d = 1$ gain agrees with recent norm-and-scale methods (Lion et al., 2026; Hübler et al., 2026; Hägele et al., 2026), yet forward-normalized Muon, with the same function class, captures only about one quarter of it (Section 4.4). Projected DGram and manifold Muon likewise show that a nominal constraint does not determine finite-budget performance independently of its optimizer. Norm and magnitude variables should be evaluated together with the state space in which momentum and preconditioning act.

Orthogonality claims are object-specific. Muon orthogonalizes an update, POET applies orthogonal transformations of a fixed random base matrix, OrthMuon constrains a directional factor whose effective rows remain orthogonal only under diagonal gains, and StelLA constrains the bases of a low-rank update. These interventions have different invariances, function classes, and induced optimizers, so evidence for one does not transfer to another; Muon's success, for example, does not predict that orthogonal weights train better. We therefore suggest nested controls: an unconstrained baseline for the full recipe, a norm-matched sphere control for pairwise angles, matched gain granularity, an expressive matrix-gain control, and optimizer-specific tuning. The target metric and numerical policy should also be explicit: POET shows that an early-training advantage need not survive a completed schedule, and POET and Pion show that a full-bfloat16 AdamW baseline can trail the same baseline in fp32 by more than the reported gain. In both the pretraining ladder and the StelLA comparison, the comparison against the unconstrained baseline is positive, while the increment from row orthogonality is small or negative.

Limitations. Appendix T discusses the scope of our experiments.

7 Conclusion

We tested the incremental value of row orthogonality while holding per-row norm treatment and gain granularity fixed. In the matched diagonal-gain ladder, $d = 8$ has no resolvable effect relative to sphere optimization, and $d = 32$ worsens mean validation loss at 360M; the per-row gap is smaller in the 1.3B and 3.135B diagnostics and grows with block size. StelLA shows the same attribution in adaptation: sphere optimization supplies the gain over Euclidean factors, while adding Stiefel geometry provides no gain.

Matrix-valued gains recover much of the large-block penalty. The $d = 32$ SPD-gain configuration has slightly lower observed loss at 1.3B and 3.135B, but this ordering is not consistent across the 360M and FPO-mix checks and does not persist beyond $d = 32$. Manifold Muon trails $d = 1$ throughout the tested 360M and 1.3B comparisons. Across the tested settings, no $d > 1$ realization provides a consistent advantage.

Orthogonal updates, latent factors, and effective weights are distinct interventions. Attributing their benefits to orthogonality calls for sphere controls with matched magnitude freedom. In our settings, the direct evidence favors sphere optimization over hard block orthogonality.

Reproducibility statement. Appendix C specifies the OrthMuon update, parameter grouping, and available implementation details; the remaining appendices report tuning grids, seed counts, per-seed

results, and paired intervals. Core code is provided upon submission, and the complete code will be provided after approval.

AI use statement. The core implementation of OrthMuon was written by the authors without generative-AI assistance. After the initial exploratory phase was completed, generative AI tools were used to help construct a fully refactored and frozen codebase for the large-scale experiments. Specifically, Claude Opus 4.8 was used to develop the implementation plan, GPT-5.6 Sol was used to write the refactored code, and Claude Opus 5.0 was used to review the resulting code. The refactored codebase was validated by reproducing a subset of experiments and confirming that the results matched those obtained with the human-written code.

Generative AI tools were also used to suggest additional ablation studies. The authors made the final decisions regarding which ablations to pursue and retained responsibility for their execution and interpretation. During manuscript preparation, GPT-5.6 Sol and Claude Code were used for grammar and language checking, cross-checking the consistency of reported experimental results, and adversarial review aimed at identifying potential errors, inconsistencies, or unsupported claims.

The authors reviewed all AI-assisted outputs and take full responsibility for the code, experimental results, claims, and final content of this work.

REFERENCES

P.-A. Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008. ISBN 978-0-691-13298-3.

Pierre-Antoine Absil and Kyle A. Gallivan. Joint diagonalization on the oblique manifold for independent component analysis. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pp. V–945–V–948, 2006. doi: 10.1109/ICASSP.2006.1661433.

Kang An, Jiaxiang Li, Donald Goldfarb, and Shiqian Ma. Demystifying manifold constraints in LLM pre-training. *arXiv preprint arXiv:2605.04418*, 2026.

Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. Can we gain more from orthogonality regularizations in training deep networks? In *Advances in Neural Information Processing Systems*, 2018.

Jeremy Bernstein. Modular manifolds. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20250926. https://thinkingmachines.ai/blog/modular-manifolds/.

Vladimir Bogachev, Vladimir Aletov, Alexander Molozhavenko, Denis Bobkov, Vera Soboleva, Aibek Alanov, and Maxim Rakhuba. RiemannLoRA: A unified riemannian framework for ambiguity-free LoRA optimization. *arXiv preprint arXiv:2507.12142*, 2025.

Kerim Büyükakyüz. OLoRA: Orthonormal low-rank adaptation of large language models. *arXiv preprint arXiv:2406.01775*, 2024.

Moustapha Cissé, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In *International Conference on Machine Learning*, 2017.

Alan Edelman, Tomás A. Arias, and Steven T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998. doi: 10.1137/S0895479895290954.

Yufei Gu and Zeke Xie. Mano: Restriking manifold optimization for LLM training. *arXiv preprint arXiv:2601.23000*, 2026.

Alexander Hägele, Alejandro Hernández-Cano, Atli Kosson, and Martin Jaggi. Improving neural network training by decoupling the magnitude and direction of weight vectors. *arXiv preprint arXiv:2606.25971*, 2026.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30016–30030, 2022.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Lei Huang, Xianglong Liu, Bo Lang, Adams Wei Yu, Yongliang Wang, and Bo Li. Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. In *AAAI Conference on Artificial Intelligence*, 2018.

Florian Hübler, Kai Lion, Antonio Orvieto, and Niao He. Muown implicitly performs angular step-size decay. *arXiv preprint arXiv:2606.23637*, 2026.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, December 2024. URL `https://kellerjordan.github.io/posts/muon/`.

Jiale Kang, Ziyin Yue, Zheng Zhan, Yangyi Huang, and Weiyang Liu. Normalized low-rank adaptation. *arXiv preprint arXiv:2608.31036*, 2026.

Jun Li, Li Fuxin, and Sinisa Todorovic. Efficient riemannian optimization on the stiefel manifold via the cayley transform. In *International Conference on Learning Representations*, 2020.

Zhizhong Li, Sina Sajadmanesh, Jingtao Li, and Lingjuan Lyu. StelLA: Subspace learning in low-rank adaptation using stiefel manifold. In *Advances in Neural Information Processing Systems*, 2025. Spotlight.

Kai Lion, Florian Hübler, Bingcong Li, Antonio Orvieto, and Niao He. Muown: Row-norm control for Muon optimization. *arXiv preprint arXiv:2605.10797*, 2026.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for LLM training. *arXiv preprint arXiv:2502.16982*, 2025.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In *International Conference on Machine Learning*, 2024a.

Weiyang Liu, Zeju Qiu, Yao Feng, Yuliang Xiu, Yuxuan Xue, Longhui Yu, Haiwen Feng, Zhen Liu, Juyeon Heo, Songyou Peng, Yandong Wen, Michael J. Black, Adrian Weller, and Bernhard Schölkopf. Parameter-efficient orthogonal finetuning via butterfly factorization. In *International Conference on Learning Representations*, 2024b.

Llama Team. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Ilya Loshchilov, Cheng-Ping Hsieh, Simeng Sun, and Boris Ginsburg. nGPT: Normalized transformer with representation learning on the hypersphere. In *International Conference on Learning Representations*, 2025.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

Juneyoung Park, Minjae Kang, Seongbae Lee, Haegang Lee, Seongwan Kim, and Jaeho Lee. Riemannian optimization for LoRA on the stiefel manifold. In *Findings of the Association for Computational Linguistics: EMNLP*, 2025.

Zeju Qiu, Weiyang Liu, Haiwen Feng, Yuxuan Xue, Yao Feng, Zhen Liu, Dan Zhang, Adrian Weller, and Bernhard Schölkopf. Controlling text-to-image diffusion by orthogonal finetuning. In *Advances in Neural Information Processing Systems*, 2023.

Zeju Qiu, Simon Buchholz, Tim Z. Xiao, Maximilian Dax, Bernhard Schölkopf, and Weiyang Liu. Reparameterized LLM training via orthogonal equivalence transformation. In *Advances in Neural Information Processing Systems*, 2025.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, 2016.

Kexuan Shi, Hanxuan Li, Zeju Qiu, Yandong Wen, Simon Buchholz, and Weiyang Liu. Pion: A spectrum-preserving optimizer via orthogonal equivalence transformation. *arXiv preprint arXiv:2605.12492*, 2026.

Mikhail Solonko, Alexander Molozhavenko, and Maxim Rakhuba. Muon on the Stiefel manifold admits an exact closed-form update. *arXiv preprint arXiv:2608.06218*, 2026.

Jianlin Su. Steepest descent on manifolds: 3. Muon + Stiefel (in Chinese). Scientific Spaces blog, `https://spaces.ac.cn/archives/11221`, August 2025.

Maksim Velikanov, Ilyas Chahed, Jingwei Zuo, Dhia Eddine Rhaiem, Younes Belkada, and Hakim Hacid. Learnable multipliers: Freeing the scale of language model matrix layers. *arXiv preprint arXiv:2601.04890*, 2026.

Eugene Vorontsov, Chiheb Trabelsi, Samuel Kadoury, and Chris Pal. On orthogonality and learning recurrent networks with long term dependencies. In *International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3570–3578, 2017.

Kaiyue Wen, Xingyu Dang, Kaifeng Lyu, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them II: Hyperball optimization. *arXiv preprint arXiv:2606.16899*, 2026a.

Kaiyue Wen, David Hall, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them. In *International Conference on Learning Representations*, 2026b. URL `https://openreview.net/forum?id=2J51qUZ0iG`.

Fei Wu, Jia Hu, Geyong Min, and Shiqiang Wang. Efficient orthogonal fine-tuning with principal subspace adaptation. In *International Conference on Learning Representations*, 2026.

Tian Xie, Haoming Luo, Haoyu Tang, Yiwen Hu, Jason Klein Liu, Qingnan Ren, Yang Wang, Wayne Xin Zhao, Rui Yan, Bing Su, Chong Luo, and Baining Guo. Controlled LLM training on spectral sphere. *arXiv preprint arXiv:2601.08393*, 2026.

Kirato Yoshihara. Different layers, different manifolds: Module-wise weight-space geometry in transformer optimization. In *ICML Workshop on Weight Space Symmetries in Deep Learning*, 2026.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. AdaLoRA: Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*, 2023.

Yuan Zhang, Jiang Hu, Zhijian Lai, Lin Lin, and Zaiwen Wen. Retraction-free optimization over the stiefel manifold for the LoRA fine-tuning. *arXiv preprint arXiv:2607.25299*, 2026.

A Extended Related Work

Orthogonality in neural-network training. Orthogonal constraints have long been used to control signal propagation, conditioning, and robustness. Parseval Networks maintain linear operators as approximate tight frames, while Orthogonal Weight Normalization enforces rectangular weight orthogonality through a differentiable reparameterization (Cissé et al., 2017; Huang et al., 2018). Softer approaches penalize deviations from orthogonality (Bansal et al., 2018), whereas Cayley-transform methods optimize parameters directly on Stiefel manifolds (Li et al., 2020). These objectives differ from spectral normalization, which controls the largest singular value without requiring weight vectors to be orthogonal to each other (Miyato et al., 2018). We revisit explicit weight orthogonality in Transformer training while separating it from per-vector norm control. Classical treatments develop optimization algorithms for Stiefel and related matrix manifolds (Edelman et al., 1998; Absil et al., 2008). In recurrent networks, soft orthogonality constraints outperformed exact constraints in the study of Vorontsov et al. (2017); our experiments revisit this question for language-model pretraining with a norm-matched sphere control.

Geometry in parameter-efficient adaptation. Orthogonality enters parameter-efficient adaptation through several distinct constructions. OFT learns an orthogonal transformation of frozen pretrained weights; BOFT represents this transformation with butterfly factors; and PSOFT restricts it to a principal subspace (Qiu et al., 2023; Liu et al., 2024b; Wu et al., 2026). These methods apply orthogonality to transformations of pretrained geometry rather than directly to effective weight rows. Within low-rank adaptation, AdaLoRA uses softly orthogonal SVD-like factors for rank allocation, and OLoRA uses an orthonormal initialization without maintaining the constraint (Zhang et al., 2023; Büyükakyüz, 2024). Recent methods optimize one or both LoRA factors on Stiefel manifolds (Park et al., 2025; Li et al., 2025), while RiemannLoRA optimizes the adapter on the fixed-rank matrix manifold to remove factorization ambiguity (Bogachev et al., 2025). Manifold-LoRA compares Stiefel and product-of-spheres constraints on a single LoRA basis (Zhang et al., 2026). Our adaptation study makes this comparison within StelLA's two-factor $USV^\top$ parameterization and separates norm control from update design.

Geometry-aware language-model pretraining. Muon orthogonalizes matrix-valued updates (Jordan et al., 2024; Liu et al., 2025; Wen et al., 2026b). Manifold Muon solves tangent-space spectral-norm steepest descent using fixed-point, dual-ascent, or exact closed-form solvers (Su, 2025; Bernstein, 2025; Solonko et al., 2026). It asks how to optimize a Stiefel-constrained weight; we ask whether that constraint improves over a sphere-constrained control with matched gain semantics. Hyperball fixes matrix Frobenius norms, while spectral-sphere optimization controls operator norms (Wen et al., 2026a; Xie et al., 2026). Mano instead constructs tangent momentum from normalized weights and alternates row- and column-wise normalization of the update while keeping the weights in Euclidean space (Gu & Xie, 2026). POET learns orthogonal transformations of a fixed random base matrix and preserves its initialized singular-value spectrum; Pion turns this isospectral structure into a direct optimizer based on left and right orthogonal transformations (Qiu et al., 2025; Shi et al., 2026). Neither method requires orthogonal effective-weight rows. Module-wise Stiefel and DGram constraints have also been studied in Transformer pretraining (Yoshihara, 2026). An et al. (2026) compare Frobenius and spectral spheres with input- and output-oblique constraints, including their interactions with RMSNorm and weight decay. Our study holds per-row magnitude freedom fixed while directly testing the addition of orthogonality between rows.

Normalization and magnitude–direction separation. Weight normalization separates a vector into a normalized direction and a learned magnitude (Salimans & Kingma, 2016). Muown is the closest prior parameterization and optimizer family to our $d = 1$ control, combining row-norm control with separately optimized magnitudes; AngularMuown relates this family to normalized directional updates (Lion et al., 2026; Hübler et al., 2026). Other recent work combines fixed-norm directions with row or column gains, while Learnable Multipliers studies the benefit of freeing scalar, row, and column scales without normalizing directions (Hägele et al., 2026; Velikanov et al., 2026). Related language-model methods normalize broader collections of parameters or representations, including nGPT (Loshchilov et al., 2025), while DoRA applies magnitude–direction decomposition to low-rank adaptation (Liu et al., 2024a). Normalized Low-Rank Adaptation (NoRA) normalizes LoRA's down-projection and also studies normalization at initialization only (Hu et al., 2022; Kang

et al., 2026). Our $d = 1$ control confirms this norm-and-scale line of work; the new comparison begins when d increases while per-row gains remain fixed.

B Experimental Protocols

Pretraining experiments apply decoupled weight decay to Muon-updated matrices scaled by the Muon learning rate, $W \leftarrow W - \eta\lambda W$ (Liu et al., 2025), use cosine decay to 0.01 times peak learning rate, and report final target-step loss on C4-val, a fixed evaluation set drawn from the C4 validation split. Hyperparameters are selected by target-step validation loss.

Table 6: Primary pretraining budgets.

Model	Budget	Steps	Train tokens
360M	2×	13,732	14.399B
360M	4×	27,464	28.798B
360M	7.69×	52,800	55.365B
1.3B	2×	48,584	50.944B
1.3B	4×	97,168	101.888B
1.3B	7.77×	188,739	197.907B
3.135B	2×	119,605	125.415B

Muon, MuonH, and the $d = 1$ control use separate brackets for the matrix or directional update learning rate; Muon and dense AdamW additionally tune weight decay in the core 360M experiments. In every Muon-family arm, embedding/output-head parameters, normalization weights, and OrthMuon gain parameters are trained with AdamW at a fixed learning rate of 0.0012, which we do not tune. The dense AdamW baseline instead tunes this as its main learning rate. The large-block SPD study in Table 26 separately selects the SPD-factor learning rate as a multiple of the shared AdamW learning rate, holding the shared AdamW and directional learning rates fixed. For each $d \in \{1, 8, 32\}$, a seed-42 directional-learning-rate bracket over $\{0.001, 0.002, 0.004\}$ selects the interior value 0.002, which is used for the confirmation seeds. Factor-of-two spacing follows standard pretraining optimizer benchmarks (Wen et al., 2026b); because every block size selects the same interior value, the bracket does not favor any d.

B.1 Model Architectures

We evaluate three sizes of a Llama-style, decoder-only causal Transformer. All models use pre-RMSNorm blocks, rotary positional embeddings (RoPE) with base $\theta = 10{,}000$, bias-free linear layers, and a SwiGLU feed-forward network,

$$\mathrm{MLP}(x) = W_{\mathrm{down}}\left[\mathrm{SiLU}(W_{\mathrm{gate}}x) \odot W_{\mathrm{up}}x\right].$$

The feed-forward dimension is three times the model width. Input and output token embeddings are tied, attention dropout is zero, and all pretraining experiments use a vocabulary of 32,000 tokens and a maximum context length of 4,096 tokens. The model configurations are summarized in Table 7.

Table 7: Model architecture hyperparameters. The GQA ratio denotes the number of query heads sharing each key–value head. Parameter counts refer to the base Transformer with tied embeddings and exclude optimizer-specific auxiliary parameters.

Model	Layers	d_{model}	d_{ff}	Q heads	KV heads	Head dim.	Parameters	Attention
360M	24	1024	3072	16	16	64	359.97M	MHA
1.3B	24	2048	6144	32	16	64	1.274B	GQA (2:1)
3.135B	32	2816	8448	22	11	128	3.135B	GQA (2:1)

The 360M model uses standard multi-head attention, with one key–value head per query head. The 1.3B and 3.135B models use grouped-query attention: each key–value head is shared by two query heads. Each Transformer layer contains the four attention projections W_Q, W_K, W_V, W_O and the three SwiGLU projections $W_{\mathrm{gate}}, W_{\mathrm{up}}, W_{\mathrm{down}}$. Although these matrices may be represented in blocks for optimization, their forward computation is identical to the corresponding dense linear projections.

C OrthMuon Implementation Details

Blockwise update. Algorithm 2 gives the update applied to each constrained weight $W \in \mathbb{R}^{m\times n}$ with row blocks of size $d \le n$. Momentum, Newton–Schulz, and the scale are computed once for the whole matrix, with G the gradient with respect to the stacked directional factors. The tangent projection and the retraction then act on each block independently, as a batched operation over the m/d blocks. For a tall matrix ($m > n$), Newton–Schulz is applied to the transpose and the result is transposed back. The five Newton–Schulz iterations are

$$X_{j+1} = \alpha X_j + \big(\beta X_j X_j^\top + \gamma (X_j X_j^\top)^2\big) X_j, \qquad (\alpha, \beta, \gamma) = (3.4445, -4.7750, 2.0315),$$

starting from $X_0 = M/\|M\|_F$.

Input: blocks $Q_k \in \mathrm{St}(d, n)$ and gain factors D_k or L_k, $k = 1, \dots, m/d$; momentum B; gradient G; learning rate η; step t; correction steps $\mathcal{T}$, 50 evenly spaced steps.

$B \leftarrow \mu B + G,\ M \leftarrow G + \mu B$	*Nesterov,* $\mu = 0.95$
$U \leftarrow 0.2\sqrt{\max(m,n)}\ \mathrm{NS}_5\big(M/\|M\|_F\big)$	*whole matrix*
for $k = 1, \dots, m/d$ **in parallel do**	
$\quad A_k \leftarrow Q_k - \eta\big(U_k - \mathrm{sym}(U_k Q_k^\top) Q_k\big)$	*tangent step*
$\quad E_k \leftarrow A_k A_k^\top - I_d$ (fp32)	*Gram error*
$\quad Q_k \leftarrow A_k + E_k\big(-\frac{1}{2} I + E_k(\frac{3}{8} I + E_k(-\frac{5}{16} I + \frac{35}{128} E_k))\big) A_k$	*Horner form*
$\quad$ **if** $t \in \mathcal{T}$ **then** $Q_k \leftarrow (Q_k Q_k^\top)^{-1/2} Q_k$	*exact polar factor*
update D_k or L_k with AdamW	*Section 3.2*

Algorithm 2: One OrthMuon step for a weight $W \in \mathbb{R}^{m\times n}$ with block size d.

Proof of Proposition 1. Tangency gives $Q\Delta^\top + \Delta Q^\top = 0$, so $AA^\top = I_m + \Delta\Delta^\top$ for $A = Q + \Delta$. Hence $E = \Delta\Delta^\top \succeq 0$ and $\|E\|_2 = \|\Delta\|_2^2 \le 1$. Because $p_r(E)$ is a polynomial in E, $Q^+Q^{+\top} = p_r(E)^2(I_m + E) = F_r(E)$ with $F_r(x) = p_r(x)^2(1 + x)$, and $\|Q^+Q^{+\top} - I_m\|_2 = \max_\lambda |F_r(\lambda) - 1|$ over the eigenvalues $\lambda \in [0, \|E\|_2]$ of E. Write $p_r(x) = \sum_{k=0}^r \beta_k x^k$ with $\beta_k = \binom{-1/2}{k}$. The function $u = (1 + x)^{-1/2}$ satisfies $2(1 + x)u' + u = 0$, so its coefficients obey $2(k + 1)\beta_{k+1} + (2k + 1)\beta_k = 0$. Truncation leaves only the top term:

$$2(1 + x)p_r'(x) + p_r(x) = (2r + 1)\beta_r x^r = -2(r + 1)\beta_{r+1} x^r.$$

Hence $F_r' = p_r\big(2(1 + x)p_r' + p_r\big) = -2(r + 1)\beta_{r+1} x^r p_r(x)$, and since $F_r(0) = 1$,

$$F_r(x) - 1 = -2(r + 1)\beta_{r+1} \int_0^x t^r p_r(t)\, \mathrm{d}t.$$

On $[0, 1]$ the series of u alternates with terms of decreasing magnitude, so its partial sums satisfy $\frac{1}{2} \le 1 - \frac{t}{2} \le p_r(t) \le 1$ for $r \ge 1$. Therefore $|F_r(x) - 1| \le 2|\beta_{r+1}| x^{r+1}$, with sign $(-1)^r$ and equality to leading order because $p_r(0) = 1$. Since $|\beta_{r+1}| = \binom{2r+2}{r+1}/4^{r+1}$, this constant is c_r, and substituting $\lambda \le \|\Delta\|_2^2$ gives the bound. For $r = 4$, the leading term is $Q^+Q^{+\top} - I_m = \frac{63}{128} E^5 + O(E^6)$.

For the cost, Horner evaluation writes $p_r(E)A = A + E\big(\beta_1 I + E(\beta_2 I + \cdots + E\beta_r)\big)A$. It needs the Gram product $AA^\top$ ($2m^2 n$ FLOPs), $r - 1$ products of $m \times m$ matrices ($2m^3$ each), and one product with A ($2m^2 n$), and it never multiplies by the identity. □

Step size and the choice $r = 4$. Newton–Schulz starts from $X_0 = M/\|M\|_F$, whose singular values lie in $[0, 1]$, and maps each of them through the same scalar quintic; on $[0, 1]$, five iterations stay below 1.203, so $\|Z\|_2 \le 1.203$ in exact arithmetic. The projection splits Z into $Z(I - Q^\top Q)$ and $\mathrm{skew}(ZQ^\top)Q$, whose rows lie in orthogonal subspaces, so $\|\Pi_Q(Z)\|_2 \le \sqrt{2}\|Z\|_2$. Hence $\|\Delta\|_2 \le 1.71\,\eta s$. By Proposition 1, the retraction error stays below the fp32 unit roundoff 2^{-24} whenever

$$\eta s \le 0.040\ (r = 2), \qquad 0.079\ (r = 3), \qquad 0.120\ (r = 4).$$

For a 1024×1024 matrix at learning rate 0.002, $\eta s = 0.0128$. Raising r from two to four costs two extra $d \times d$ products per block, $4md^2$ FLOPs per matrix, but triples the step size covered by this guarantee. The implementation originally chose $r \in \{2, 3, 4\}$, or an exact polar step, from $\|E\|_F$. That choice reads a scalar back to the host at every step, which synchronizes the GPU with the CPU and breaks graph compilation. Fixing $r = 4$ removes the synchronization, and the projection and retraction compile into one graph.

Implementation of the retraction. The polar factor of $p_r(E)A$ equals that of A, because $p_r(E)$ is symmetric positive definite and commutes with $AA^\top$. An exact correction applied after the polynomial therefore yields exactly the polar retraction of the tangent step. When the parameters are stored in bfloat16, the Gram error and the polynomial are computed in fp32 and the result is cast back. Exact corrections compute the inverse square root from an eigendecomposition of $QQ^\top$, with an SVD fallback. They occur at 50 evenly spaced steps regardless of the training length.

Arithmetic cost. Counting matrix-multiply FLOPs, one Newton–Schulz iteration on an $m \times n$ matrix with $m \le n$ costs $4m^2n + 2m^3$, so five iterations cost $20m^2n + 10m^3$. For one $d \times n$ block, the tangent projection costs $4d^2n$ (the products $UQ^\top$ and $\mathrm{sym}(\cdot)Q$), and the order-4 retraction costs $4d^2n + 6d^3$ by Proposition 1. Over all m/d blocks, the added cost is $8mdn + 6md^2$, at most $0.6\,d/m$ of the Newton–Schulz cost when $m \le n$. For $m > n$, the same count gives at most $0.7\,d/n$. For a 1024×1024 projection with $d = 32$, the added cost is 0.9% of Newton–Schulz. Measured throughput shows no growth with d (Table 8): with per-row gains, OrthMuon is 0.2–1.1% slower than Muon at 360M and 2.8–3.0% slower at 1.3B.

Table 8: Training throughput (thousand tokens/s) on 8 H100 GPUs, $2\times$ budget: median instantaneous throughput over steps 1,000–11,320. Change is relative to Muon at the same scale.

	360M		1.3B	
Method	k tokens/s	Change	k tokens/s	Change
Muon	976.2	–	377.8	–
OrthMuon $d = 1$	965.3	-1.1%	366.6	-3.0%
OrthMuon $d = 8$	965.6	-1.1%	–	–
OrthMuon $d = 32$	974.6	-0.2%	367.4	-2.8%

Parameters and initialization. The constrained matrices are the q, k, v, o projections and the MLP gate, up, and down projections. Embeddings, embedding-tied output parameters, and normalization weights remain in AdamW. Matrix rows are output channels and are grouped contiguously; a row count not divisible by d raises an error rather than creating a remainder block. Each directional factor is initialized by orthonormalizing a standard random initialization with a reduced QR decomposition. Matrix-gain runs initialize the effective left gain to $S_k = I_d$. Row gains and L_k are trained with AdamW at the shared AdamW-side learning rate 0.0012 and zero weight decay; deviations are stated with the corresponding ablation. Constrained directional factors receive no weight decay.

Manifold-Muon solvers. At full width (d equal to the number of matrix rows), manifold Muon uses the exact square-case update $\Phi = Q\,\mathrm{msign}(\mathrm{skew}(Q^\top G))$, where $\mathrm{skew}(A) = \frac{1}{2}(A - A^\top)$ and $\mathrm{msign}(A) = UV^\top$ for an SVD $A = U\Sigma V^\top$. For $d \le 256$, it instead uses a thin-QR reduction to a $2d \times 2d$ matrix. The gain parameterization and retraction are unchanged.

Direction–magnitude coordinates. Even when two parameterizations represent the same effective weights, they need not induce the same update. For one row, write $w = aq$ with $\|q\|_2 = 1$. An infinitesimal effective update decomposes as

$$\mathrm{d}w = q\,\mathrm{d}a + a\,\mathrm{d}q, \qquad q^\top \mathrm{d}q = 0. \tag{6}$$

The gain controls both the effective row norm and the conversion between a step in q and an angular step in w.

Polar expressivity. For a full-row-rank block W_k, the matrix $P_k = (W_k W_k^\top)^{1/2}$ is positive definite, and

$$W_k = P_k Q_k, \qquad P_k = (W_k W_k^\top)^{1/2}, \qquad Q_k = P_k^{-1} W_k, \tag{7}$$

where $Q_k Q_k^\top = I_d$. Thus a positive-definite matrix gain can represent any full-row-rank effective block even though the latent directional factor remains orthogonal.

D Other Optimizers and Controls

Manifold Muon. Manifold Muon solves spectral-norm steepest descent directly inside the Stiefel tangent space (Bernstein, 2025; Su, 2025; Solonko et al., 2026), with the same magnitude parameterization and retraction as OrthMuon; Appendix C gives the full-width and thin-QR solvers.

Forward normalization. The same effective function class need not induce the same update. For $w = aq$, the gain couples a step in q to the effective angular step (Appendix C). Forward normalization stores momentum on an unconstrained latent weight, applies Muon to it, and differentiates through row normalization; it preserves the $d = 1$ function class but changes the optimization trajectory. The default OrthMuon stores momentum in ambient coordinates; tangent and transported-momentum variants serve as further controls (Appendix O).

No-gain control. Setting $D_k = I_d$ removes learned magnitudes. At $d = 1$, this fixes every row norm without constraining angles between rows. Dense Muon, no-gain $d = 1$, and learned-gain $d = 1$ form our norm-and-magnitude ablation.

MuonH/Hyperball. MuonH constrains the Frobenius norm of each full matrix to its initial value, explicitly normalizes its update, and uses no learned gain or weight decay (Wen et al., 2026a). It is a coarser norm-control baseline with no pairwise orthogonality.

Other left factors. Besides the diagonal and SPD factors of Section 3.2, we train eight other left factors $W_k = F_k Q_k$, all initialized at $F_k = I_d$: $F_k = X_k X_k^\top$ with a full square X_k (XXT); $\mathrm{diag}(a_k) + X_k X_k^\top$, whose X_k is updated with Muon; the symmetric square of a symmetric matrix, also updated with Muon; $L_k L_k^\top + \mathrm{diag}(\lambda_k)$; the matrix exponential $\exp(A_k)$ of a symmetric A_k; an indefinite symmetric matrix $\mathrm{diag}(a_k) + \mathrm{Off}(M_k)$; a lower-triangular T_k with diagonal $\mathrm{diag}(a_k)$, which gives the LQ form $W_k = T_k Q_k$; and a singular-value form $U_k \,\mathrm{diag}(s_k)$, whose orthogonal U_k is updated with the same Stiefel update as Q_k. Except where noted, the factors are trained with AdamW. Every one of them can represent any full-row-rank block. Results are in Appendix K.

Table 9: The seventeen orthogonality-training variants and where their results appear.

	Variant	Results
OrthMuon with block-Stiefel directions; left factor F_k in $W_k = F_k Q_k$		
1	Per-row gain $\mathrm{diag}(a_k)$	Table 1, Figure 1
2	One scalar gain per block	Table 23
3	SPD $L_k L_k^\top$	Tables 23, 25
4	$X_k X_k^\top$	Tables 23, 25
5	$\mathrm{diag}(a_k) + X_k X_k^\top$, X_k updated with Muon	Table 23
6	Symmetric square, updated with Muon	Table 24
7	$L_k L_k^\top + \mathrm{diag}(\lambda_k)$	Table 25
8	Matrix exponential $\exp(A_k)$	Table 25
9	Indefinite symmetric $\mathrm{diag}(a_k) + \mathrm{Off}(M_k)$	Table 25
10	Lower triangular T_k	Table 25
11	Singular-value $U_k \,\mathrm{diag}(s_k)$	Table 25
Alternative updates for $d = 1$		
12	Direct row normalization	Table 36
13	Tangent gradient before momentum	Table 36
14	Tangent gradient with transported momentum	Table 36
Other optimizers for the same constraint		
15	Manifold Muon, per-row gain	Table 28
16	Manifold Muon, SPD gain	Table 28
17	Projected diagonal-Gram (DGram)	Table 23; App. M

E ADAPTATION SETUP

For a base weight $W_0 \in \mathbb{R}^{p\times q}$, StelLA writes the rank-r update as (Li et al., 2025)

$$\Delta W = \frac{\alpha}{r} USV^\top, \qquad U \in \mathbb{R}^{p\times r},\ S \in \mathbb{R}^{r\times r},\ V \in \mathbb{R}^{q\times r}. \tag{8}$$

The rank directions are the columns of U and V. Euclidean factors are unconstrained. Sphere optimizes each rank direction independently on a unit sphere while allowing arbitrary pairwise angles. Stiefel additionally requires $U^\top U = V^\top V = I_r$. All three use the same learned S, making Stiefel versus Sphere the direct comparison for adding pairwise angle constraints at fixed factor norms. The canonical Stiefel update tangent-projects both the ambient gradient and the post-Adam displacement before polar retraction; the direct-polar control omits only the post-Adam tangent projection.

The adaptation study uses Llama-3-8B with rank 32, $\alpha = 64$, and adapters on the query, key, value, up-projection, and down-projection matrices (Llama Team, 2024). We train for three epochs with global batch 16, learning rate 5×10^{-4}, linear decay, and 500 warmup steps. All main-table arms use StelLA's default learning rate 5×10^{-4} and $S_0 = I$; each run loads the checkpoint with the lowest token-level loss on a fixed 120-example validation split. The learning-rate and decay grids in Appendix R are sensitivity checks, not selection rules for the main table. Scores are equal-weight macro accuracy over BoolQ, PIQA, SocialIQA, HellaSwag, WinoGrande, ARC-Easy, ARC-Challenge, and OpenBookQA.

F PER-SEED MATCHED ORTHOGONALITY LADDERS

Table 10: Paired change in C4 validation loss from adding row orthogonality, relative to the $d = 1$ control, in milli-loss (positive means worse). Each difference compares the same seed; intervals are paired 95% t-intervals, and the final column counts higher-loss $d > 1$ runs.

Budget	Contrast	Mean Δ	95% interval	Worse seeds
2×	$d = 8 - d = 1$	+0.41	$[-3.52, 4.34]$	2/3
2×	$d = 32 - d = 1$	+5.45	$[1.62, 9.27]$	3/3
7.69×	$d = 8 - d = 1$	+0.53	$[-3.42, 4.49]$	2/3
7.69×	$d = 32 - d = 1$	+4.67	$[1.88, 7.46]$	3/3

Table 11: Per-seed final C4 loss for the matched per-row-gain ladder.

Budget	d	Seed 42	Seed 1234	Seed 5678	Mean ± SD
2×	1	2.4640	2.4621	2.4612	2.4624 ± 0.0015
2×	8	2.4626	2.4631	2.4627	2.4628 ± 0.0003
2×	32	2.4681	2.4671	2.4683	2.4679 ± 0.0006
7.69×	1	2.3576	2.3567	2.3558	2.3567 ± 0.0009
7.69×	8	2.3566	2.3570	2.3580	2.3572 ± 0.0007
7.69×	32	2.3614	2.3609	2.3617	2.3614 ± 0.0004

All rows in Table 11 belong to the matched d-ladder cohort and use one learned gain per weight row. Main-text differences are computed from the unrounded values. The paired $d = 32 - d = 1$ effects are $+0.00545$ at 2× (95% t-interval $[0.00162, 0.00927]$) and $+0.00467$ at 7.69× ($[0.00188, 0.00746]$). The $d = 8 - d = 1$ signs are mixed at both budgets.

G SPHERE OPTIMIZATION AND LEARNED MAGNITUDE

Table 12: Final C4 validation loss at 360M.

Method	2×	4×	7.69×
Tuned Muon	2.475	2.418	2.372
MuonH	2.469	2.409	2.363
$d = 1$, no gain	2.468	2.409	2.365
$d = 1$, row gain	**2.462**	**2.401**	**2.357**

Table 13: Loss improvements over tuned Muon at 360M, computed from unrounded means; positive values indicate lower loss.

Component	$2\times$	$4\times$	$7.69\times$
Unit-row direction constraint, no gain	0.0068	0.0091	0.0070
Learned row magnitude	0.0067	0.0078	0.0082
Total $d = 1$ improvement	0.0135	0.0169	0.0151

MuonH and no-gain $d = 1$ differ by only 0.0010, 0.0004, and -0.0023 loss across the three budgets, despite constraining one norm per matrix versus one norm per row.

H Scaling of the $d = 1$ Control

Table 14: Scaling of the $d = 1$ control on C4.

Model	Budget	Muon	MuonH	$d = 1$
1.3B	$2\times$	2.1978	2.1912	**2.1846**
1.3B	$4\times$	2.1527	2.1466	**2.1406**
1.3B	$7.77\times$	2.1203	2.1139	**2.1073**
3.135B	$2\times$	2.0365	2.0309	**2.0246**

Table 15: Larger-scale $d = 32$ diagnostics on C4. Deltas are relative to the $d = 1$ result at the same model size.

Model	d / gain	Loss	$\Delta \times 10^3$
1.3B	$d = 1$, per row	2.1846	0.0
1.3B	$d = 32$, per row	2.1864	+1.8
1.3B	$d = 32$, SPD	2.1843	-0.4
3.135B	$d = 1$, per row	2.0246	0.0
3.135B	$d = 32$, per row	2.0249	+0.3
3.135B	$d = 32$, SPD	2.0245	-0.1

I FPO-Mix Training-Mixture Check

FPO-mix denotes the pretraining mixture of Wen et al. (2026b). It contains 54.401B tokens, with realized proportions 92.57% DCLM, 6.09% StarCoder, and 1.34% ProofPile. Training-mixture comparisons use C4-val.

Table 16: FPO-mix learning-rate sweep at 360M, $2\times$.

Method	LR	Final C4 loss
MuonH	0.00625	2.7037
MuonH	0.0125	**2.6764**
MuonH	0.025	2.7152
OrthMuon $d = 1$	0.001	2.7085
OrthMuon $d = 1$	0.002	**2.6706**
OrthMuon $d = 1$	0.004	2.6818

Table 17: FPO-mix results at 360M, $2\times$, evaluated on C4-val.

Method	Mean loss $\pm$ SD
OrthMuon $d = 1$	**2.6701** $\pm$ 0.0007
OrthMuon $d = 32$, per-row gain	2.6759 $\pm$ 0.0005
MuonH	2.6769 $\pm$ 0.0013
Muon	2.6818 $\pm$ 0.0003

The Muon row in the preceding three-seed table uses learning rate 0.004 and weight decay 0.3, selected by a seed-42 bracket over learning rates $\{0.002, 0.004, 0.008\}$.

Table 18: Separate 360M FPO-mix reproduction at the shared AdamW-side learning rate 0.0012, evaluated on C4-val.

Budget	Method	Directional LR	Final loss
$2\times$	$d = 1$	0.002	2.669971
$2\times$	$d = 32$ SPD	0.001	2.705358
$2\times$	$d = 32$ SPD	0.002	2.669714
$2\times$	$d = 32$ SPD	0.004	2.679970
$4\times$	$d = 1$	0.002	2.607065
$4\times$	$d = 32$ SPD	0.001	2.626027
$4\times$	$d = 32$ SPD	0.002	2.606911
$4\times$	$d = 32$ SPD	0.004	2.623826

At the common directional learning rate 0.002, $d = 1$ and $d = 32$ SPD differ by 0.257 milli-loss at $2\times$ and 0.154 milli-loss at $4\times$. These are training-mixture checks; the reported metric remains C4 validation loss.

Table 19: FPO-mix 1.3B, $2\times$, evaluated on C4-val.

Method	Selected LR / WD	Mean loss $\pm$ SD
OrthMuon $d = 1$	0.001	**2.4010** $\pm$ 0.0003
MuonH	0.008	2.4076 $\pm$ 0.0011
Muon	0.002 / 0.3	2.4128 $\pm$ 0.0004

J Alternate-Corpus Evaluation

We evaluate the final 360M C4-trained checkpoints at $2\times$ and $7.69\times$ on six alternate-corpus sets without further training, at sequence length 4096, using 12.6M tokens per FPO set and 14.7M tokens per OLM set. Every checkpoint was also replayed on C4; the maximum deviation from its recorded loss was 0.0019 at $2\times$ and 0.0021 at $7.69\times$. The latter is above the original 0.002 threshold and within the evaluator's explicit 0.0025 tolerance. C4 removed every page containing a curly bracket (Raffel et al., 2020), which biases it away from code- and LaTeX-heavy pages. We therefore report the predominantly natural-language sets (FPO mixture, DCLM, OLM-Wiki) separately from StarCoder, ProofPile, and OLM-arXiv. Single-seed tuned arms are omitted.

At $2\times$ (Table 20), $d = 32$ has the lowest mean loss on the natural-language-heavy sets, but its seed standard deviation is 0.012–0.017, compared with 0.002–0.004 for $d = 1$. Against Muon, $d = 32$ is lower on four of six sets for seeds 42 and 1234 and on none for seed 5678. On the remaining sets, Muon has the lowest mean loss, and seed variation is much larger than on the natural-language-heavy sets.

Table 21 compares all methods on the two common seeds; Table 22 gives the three-seed ladder, which the main text uses for d-ladder differences. Adding seed 5678 does not change the best d on any of the six alternate-corpus sets: $d = 1$ is lowest on the FPO mixture and DCLM, $d = 8$ on StarCoder, ProofPile, and OLM-arXiv, and $d = 32$ on OLM-Wiki. The observed ordering varies with evaluation domain and training budget.

Table 20: Mean loss of 360M, $2\times$, C4-trained checkpoints on alternate-corpus sets (sample standard deviation in parentheses. Lowest mean per column in bold.

		Natural-language-heavy			Code/math/LaTeX-rich	
Method	FPO mix	DCLM	OLM-Wiki	StarCoder	ProofPile	OLM-arXiv
AdamW	3.0932 (.0067)	3.0932 (.0084)	3.2224 (.0266)	3.0470 (.0565)	3.9833 (.0364)	2.8362 (.0490)
Muon	3.0170 (.0044)	3.0190 (.0031)	3.1361 (.0021)	**2.9682** (.0269)	**3.7769** (.0152)	**2.7263** (.0160)
MuonH	3.0316 (.0128)	3.0288 (.0081)	3.1505 (.0070)	3.0363 (.0544)	3.8737 (.0447)	2.7694 (.0349)
$d = 1$, no gain	3.0378 (.0060)	3.0315 (.0035)	3.1651 (.0048)	3.0755 (.0859)	3.8629 (.0154)	2.7763 (.0497)
$d = 1$	3.0238 (.0040)	3.0185 (.0025)	3.1328 (.0019)	3.0607 (.0217)	3.8132 (.0402)	2.7485 (.0337)
$d = 8$, per row	3.0167 (.0060)	3.0125 (.0019)	3.1336 (.0044)	3.0223 (.0484)	3.8709 (.0490)	2.7734 (.0118)
$d = 32$, per row	**3.0132** (.0170)	**3.0097** (.0117)	**3.1277** (.0164)	3.0139 (.0852)	3.8579 (.0673)	2.7430 (.0360)

Table 21: Mean loss of 360M, $7.69\times$, C4-trained checkpoints on alternate-corpus sets, (sample standard deviation in parentheses). Lowest mean per column in bold.

		Natural-language-heavy			Code/math/LaTeX-rich		
Method	C4	FPO mix	DCLM	OLM-Wiki	StarCoder	ProofPile	OLM-arXiv
Muon	2.3723	2.9552 (.0054)	2.9500 (.0125)	3.0198 (.0066)	2.9809 (.0590)	3.7199 (.0017)	2.6603 (.0072)
MuonH	2.3631	2.9313 (.0156)	2.9344 (.0130)	3.0644 (.0192)	2.8689 (.0404)	3.7366 (.0213)	2.6870 (.0124)
$d = 1$	2.3572	**2.9163** (.0114)	**2.9225** (.0133)	3.0365 (.0114)	2.8059 (.0080)	3.7075 (.0135)	2.6871 (.0157)
$d = 8$, per row	**2.3568**	2.9260 (.0136)	2.9357 (.0133)	3.0234 (.0132)	**2.7775** (.0176)	**3.6909** (.0241)	**2.6416** (.0233)
$d = 32$, per row	2.3612	2.9354 (.0214)	2.9352 (.0242)	**3.0097** (.0113)	2.9023 (.0031)	3.7233 (.0018)	2.6441 (.0093)

Table 22: Three-seed OrthMuon ladder at 360M, $7.69\times$, on alternate-corpus sets (seeds 42, 1234, 5678; sample standard deviation in parentheses). Lowest mean per column in bold.

		Natural-language-heavy			Code/math/LaTeX-rich		
Method	C4	FPO mix	DCLM	OLM-Wiki	StarCoder	ProofPile	OLM-arXiv
$d = 1$	**2.3567**	**2.9179** (.0085)	**2.9237** (.0096)	3.0471 (.0201)	2.8155 (.0175)	3.7124 (.0128)	2.6954 (.0182)
$d = 8$, per row	2.3572	2.9297 (.0115)	2.9384 (.0105)	3.0251 (.0097)	**2.8065** (.0518)	**3.6745** (.0332)	**2.6398** (.0168)
$d = 32$, per row	2.3614	2.9375 (.0156)	2.9398 (.0188)	**3.0175** (.0157)	2.8815 (.0360)	3.7288 (.0096)	2.6655 (.0376)

K MATRIX-GAIN CONTROLS

Table 23: Selected $d > 1$ controls at $d = 32$ unless noted and Δ is in milli-loss relative to the panel's $d = 1$ reference. SPD, XXT, and XXT+diagonal use full-matrix block gains. Large-block and manifold-Muon results appear in the tables below.

Realization	Loss	Δ
360M $d = 1$	2.4617	-
Stiefel, per-block gain	2.4693	+7.6
SPD $LL^\top Q$	2.4627	+1.0
XXT	2.4631	+1.4
XXT + diagonal	2.4621	+0.4
Projected DGram	2.4808	+19.1

Realization	Loss	Δ
1.3B $d = 1$	2.1846	-
Stiefel, per-row gain	2.1864	+1.8
SPD	2.1843	-0.4
SPD, $d = 64$	2.1851	+0.5
3.135B $d = 1$	2.0246	-
Stiefel, per-row gain	2.0249	+0.3
SPD	2.0245	-0.1

Table 24: Complementary 360M, $2\times$ matrix-gain study.

Parameterization	Gain granularity	Loss	Δ vs. $d = 1$
$d = 1$	per row	2.4617	0.0000
$d = 32$, scalar	per block	2.4693	+0.0076
$d = 32$, SPD $LL^\top$	full matrix	2.4627	+0.0010
$d = 32$, XXT	full matrix	2.4631	+0.0014
$d = 32$, XXT + diagonal	full matrix	2.4621	+0.0004
$d = 32$, symmetric-square + Muon	full matrix	2.4631	+0.0014

Table 25: Left factors at $d = 32$ in an earlier 360M, $2\times$ cohort (three seeds each; Δ in milli-loss from the cohort's $d = 1$, 2.4494).

Left factor	Form	Loss	Δ
Per-row	$\mathrm{diag}(a)$	2.4555	+6.1
SPD	$LL^\top$	2.4499	+0.5
SPD + diagonal	$LL^\top + \mathrm{diag}(\lambda)$	2.4506	+1.2
XXT	$XX^\top$	2.4506	+1.2
Matrix exponential	$\exp(A)$, $A = A^\top$	2.4510	+1.6
Indefinite symmetric	$\mathrm{diag}(a) + \mathrm{Off}(M)$	2.4513	+1.9
Lower triangular	T	2.4525	+3.1
Singular-value	$U\,\mathrm{diag}(s)$, $U^\top U = I$	2.4553	+5.9

Table 25 comes from an earlier cohort whose cosine schedule decays to $0.1\times$ rather than $0.01\times$ the peak learning rate and whose validation subset has 0.79M tokens rather than 16M, so only differences within the table are comparable. On this subset, differences below about 1.5 milli-loss are not resolved, and the symmetric factors are indistinguishable from one another. The lower-triangular factor trails SPD by 2.6 milli-loss, and the singular-value form recovers almost none of the per-row penalty. For the singular-value form, frame learning rate 0.0005 improves on 0.002 by 0.7 milli-loss; the table reports 0.0005.

Table 26: Selected seed-42 large-block results at 360M. Each method uses its selected main learning rate; SPD gain learning rates are selected separately.

	$2\times$		$4\times$	
d	Per-row gain	SPD gain	Per-row gain	SPD gain
128	2.4822	2.4685	2.4168	2.4061
256	2.5005	2.4790	2.4345	2.4191
1024	2.6185	2.5458	2.5522	2.4915

Across both budgets, SPD recovers an increasing absolute amount of the large-block penalty, while all six selected SPD cells remain worse than the corresponding $d = 1$ control. The $4\times$, $d = 1024$ per-row-gain selection lies at the upper edge of its tested main-learning-rate bracket.

Table 27: Selected seed-42 large-block results at 1.3B, $2\times$. Every directional-learning-rate bracket selects the interior value 0.001.

d	Per-row gain	SPD gain	$\Delta_{\mathrm{row}} \times 10^3$	$\Delta_{\mathrm{SPD}} \times 10^3$
128	2.1927	2.1891	+8.0	+4.5
256	2.1992	2.1973	+14.6	+12.7
1024	2.2409	2.2544	+56.2	+69.8

The columns Δ_{row} and Δ_{SPD} in Table 27 are differences of the per-row-gain and SPD-gain arms from the three-seed $d = 1$ mean 2.1846. SPD recovers part of the large-block penalty at $d = 128$ and 256, but is worse than the per-row-gain arm at $d = 1024$.

Table 28: Selected 360M manifold-Muon results, "–" denotes a missing same-d OrthMuon reference.

Budget	d	Gain	Manifold Muon	OrthMuon same d	$d = 1$
$2\times$	32	per row	2.4945	2.4681	2.4640
$2\times$	32	SPD	2.4919	2.4607	2.4640
$2\times$	256	per row	2.5060	2.5005	2.4640
$2\times$	1024	per row	2.6037	2.6185	2.4640
$4\times$	32	per row	2.4257	–	2.4011
$4\times$	32	SPD	2.4237	–	2.4011
$4\times$	256	per row	2.4390	2.4345	2.4011
$4\times$	1024	per row	2.5404	2.5522	2.4011

Among cells with an available same-d reference, manifold Muon improves on projected OrthMuon only at $d = 1024$, while remaining far behind $d = 1$ throughout. At 1.3B (Table 29), it trails projected OrthMuon at every tested block size: by 40.9, 54.7, 18.1, and 5.4 milli-loss for per-row $d = 32$, SPD $d = 32$, per-row $d = 256$, and per-row $d = 1024$. All four arms are best at the smallest tested learning rate, 0.001, which is also the rate selected for OrthMuon at 1.3B.

Table 29: 1.3B, $2\times$ manifold-Muon runs, best of learning rates $\{0.001, 0.002, 0.004\}$. Δ is relative to the three-seed $d = 1$ mean 2.1846.

d	Gain	Manifold Muon	OrthMuon same d	Δ vs. $d = 1$ ($\times 10^{-3}$)
32	per row	2.2273	2.1864	+42.7
32	SPD	2.2380	2.1843	+53.4
256	per row	2.2173	2.1992	+32.7
1024	per row	2.2463	2.2409	+61.7

L Row Geometry of Trained Checkpoints

We analyze the final seed-42 checkpoints of $d = 1$, per-row $d = 32$, SPD $d = 32$, tuned Muon, and MuonH at 360M, 2×. The $d = 1$ checkpoint uses gain learning rate 0.0048 (Appendix N). Every matrix is split into the same contiguous 32-row windows in its stored-row orientation, so the $d = 1$ Gram matrices are not trivially 1×1. In this orientation every stored row lies in $\mathbb{R}^{1024}$, including the MLP down projection, which is stored as a 3072×1024 matrix. Statistics are pooled over all 24 layers. For the effective rows w_i of a window, the cosine is $C_{ij} = \langle w_i, w_j \rangle / (\|w_i\|_2 \|w_j\|_2)$. Each arm contributes 9,984 windows and about 4.95M row pairs. For independent isotropic rows of width 1024, the expected absolute cosine is $\sqrt{2/(1024\pi)} \approx 0.025$, and blocks of 32 random unit rows have median cosine-matrix largest eigenvalue 1.34 and median condition number 1.38 (4,000 simulated blocks). Stable rank is $\|W\|_F^2 / \|W\|_2^2$ of a 32-row block.

Table 30: Geometry of effective 32-row blocks pooled over all matrices. Mean and p95 are over row pairs; the other columns are medians over blocks.

Arm	Mean $\lvert C_{ij} \rvert$	p95 $\lvert C_{ij} \rvert$	$\lambda_{\max}(C)$	Stable rank	Condition number
Random unit rows	0.025	–	1.34	23.9	1.38
Muon	0.033	0.082	1.385	20.4	1.62
MuonH	0.033	0.084	1.386	20.5	1.62
$d = 1$	0.030	0.076	1.378	19.2	1.74
Per-row $d = 32$	$< 10^{-8}$	$< 10^{-7}$	1.000	25.3	1.31
SPD $d = 32$	0.052	0.142	1.774	17.0	1.94

Table 31: Absolute cosine between effective rows in the same 32-row window. Per-row $d = 32$ is below 10^{-8} in every matrix type.

	Muon		$d = 1$		SPD $d = 32$	
Matrix	Mean	p95	Mean	p95	Mean	p95
Query	0.054	0.172	0.046	0.137	0.105	0.326
Key	0.056	0.168	0.046	0.130	0.111	0.323
Value	0.047	0.169	0.032	0.091	0.047	0.130
Attention output	0.029	0.072	0.028	0.068	0.046	0.116
MLP gate	0.026	0.066	0.026	0.064	0.044	0.112
MLP up	0.026	0.064	0.026	0.063	0.039	0.097
MLP down	0.027	0.066	0.026	0.065	0.038	0.096
All	0.033	0.082	0.030	0.076	0.052	0.142

Trained $d = 1$ rows stay close to the random scale in the MLP but become more coherent in query and key matrices. Unconstrained Muon rows are slightly more coherent than $d = 1$ rows in query, key, and value matrices and match them elsewhere; MuonH is within 0.001 of Muon in pooled mean absolute cosine. Pooled over all matrices, the $d = 1$ directional factor is close to the random level: its median block condition number is 1.42, against 1.38 for random unit rows. The per-row $d = 32$ constraint removes this coherence exactly, and the SPD gain restores and amplifies it while its directional factor Q remains orthogonal; the increase is largest in query and key matrices. Condition numbers show the same pattern. The median condition number of an effective 32-row window is 5.5 and 4.9 for $d = 1$ key and query matrices, 2.5 and 2.2 for per-row $d = 32$, and 8.0 and 7.2 for SPD $d = 32$. Blockwise orthogonality does not make whole matrices well conditioned: the median condition number of the 1024×1024 query and key matrices is about 2.5×10^4 for $d = 1$ and 1.3–1.4 $\times 10^4$ for per-row $d = 32$. Per-row $d = 32$ improves block conditioning by construction and also has the highest stable rank (Table 30), yet it has higher loss than $d = 1$. Tuned Muon also has better-conditioned, higher-stable-rank blocks than $d = 1$ and higher loss (2.475 versus 2.462; Table 1). Better conditioning and higher stable rank therefore do not translate into lower validation loss.

Table 32: Geometry of effective 128- and 1024-row windows pooled over all matrices. Mean is over row pairs; the other columns are medians over windows. Pooled condition numbers of 1024-row windows mix matrix types and are omitted.

Window	Arm	Mean $\lvert C_{ij}\rvert$	$\lambda_{\max}(C)$	Stable rank	Condition number
128	Random unit rows	0.025	1.80	71.1	2.04
	Muon	0.030	1.99	58.5	2.47
	$d = 1$	0.028	1.94	53.3	2.66
	Per-row $d = 32$	0.021	1.72	68.8	2.14
	SPD $d = 32$	0.035	2.45	48.6	2.86
1024	Random unit rows	0.025	3.94	260	–
	Muon	0.028	6.87	147.7	–
	$d = 1$	0.027	6.52	146.6	–

Larger windows give the same ordering (Table 32). In 128-row windows, 1,984 of the 8,128 row pairs lie in the same $d = 32$ block, so the across-block pairs of per-row $d = 32$ average 0.027, close to 0.028 for $d = 1$ rows; block orthogonality does not propagate beyond the block. In 1024-row windows, $d = 1$ and Muon have nearly the same spectrum, with largest cosine-matrix eigenvalues 6.5 and 6.9 against 3.9 for random rows and stable ranks 147 and 148 against 260. This large-scale concentration therefore arises without any constraint, and the sphere constraint does not change it.

M Projected DGram

Table 33: Projected DGram $d = 32$ LR–WD sweep at 360M, 2×. Entries are three-seeds means.

LR	WD 0.1	WD 0.3	WD 1.0
0.002	2.5147	2.4899	2.4924
0.004	2.5021	**2.4808**	2.5062
0.008	2.8249	2.5093	2.5478

Table 34: Projected DGram block-size screen without weight decay.

Block size	Selected LR	Final loss
32	0.002	2.5452
128	0.002	2.5592
1024	0.004	2.6908

N Module-Wise Substitution

We reprodce this work using the official code [1]. Table 35 tests module assignment at 360M, 2×, with three seeds. One module type, attention (q, k, v, o) or MLP (gate, up, down), stays under $d = 1$ with per-row gains, directional learning rate 0.002, weight decay 0.1, and gain learning rate 0.0048, four times the shared AdamW learning rate of 0.0012, which is unchanged. The other module type is replaced. The reference trains every matrix with these $d = 1$ settings and reaches 2.4598. Replacement settings are the tuned settings of each method: Muon uses learning rate 0.004 and weight decay 0.3; MuonH uses 0.0125 and no weight decay; per-row and SPD $d = 32$ use directional learning rate 0.002, weight decay 0.1, and gain learning rate 0.0012, with no weight decay on the SPD factor; projected DGram uses $d = 32$, learning rate 0.004, and weight decay 0.3. The replaced $d = 32$ modules thus use a smaller gain learning rate than the fixed $d = 1$ modules and the reference.

[1] Module-Wise Muon: https://github.com/kiratoyoshihara/module-wise-manifold-muon.

Table 35: Module-wise substitution at 360M, 2×: three-seeds mean final C4 loss and change from the all-$d = 1$ reference 2.4598 ($\times 10^{-3}$; negative is better).

	MLP replaced		Attention replaced	
Replacement	Loss	Δ	Loss	Δ
Muon	2.4713	+11.5	2.4675	+7.7
MuonH	2.4624	+2.6	2.4679	+8.1
Per-row $d = 32$	2.4629	+3.1	2.4642	+4.4
SPD $d = 32$	2.4622	+2.4	2.4583	−1.5
Projected DGram	2.4732	+13.4	2.4708	+11.0

Projected DGram is the worst replacement in either module, and it is worst in MLP matrices, where Yoshihara (2026) assign it. Per-row $d = 32$ is worse than $d = 1$ in both module types. MuonH is close to $d = 1$ in MLP matrices but not in attention, whereas Muon trails in both. SPD $d = 32$ in attention is the only replacement below the reference, by less than the seed-to-seed spread of $d = 1$. Each replacement prefers the same module type in all seeds: MuonH and per-row $d = 32$ do better in MLP matrices, and Muon, SPD $d = 32$, and projected DGram do better in attention matrices. The three seeds of each cell differ by at most 1.8 milli-loss.

O Additional Optimizer and Coordinate Controls

Table 36: Selected 360M, 2×, $d = 1$ controls.

Variant	n	Final loss
Default OrthMuon	3	2.4617
Direct row normalization	2	2.4632
Exponential gain	3	2.4622
Tangent gradient before momentum	3	2.4641
Tangent + transported momentum	3	2.4630
Block-Frobenius tangent $d = 32$ + row gain	1	2.4653
Forward-normalized Muon	3	2.4715
Polar-split update only	3	2.5679

The displayed forward-normalized result uses LR 0.002, WD 1.0, and three seeds. The lowest grid mean, at LR 0.004 and WD 0.3, is 2.471443 over three seeds and differs by less than 10^{-5}. The block-Frobenius tangent control constrains only the norm of each 32-row block, not pairwise angles; its best tested LR, 0.016, is the upper edge of the two-point bracket $\{0.008, 0.016\}$. The polar-split control applies the polar decomposition to the Muon update without maintaining a constrained weight state.

P POET Comparison Details

We reprodce POET using the official code [2]. OrthAdam $d = 1$ uses a unit-row sphere direction and one learned scalar per row, with AdamW rather than Muon on the constrained matrix roles. The POET construction combines two orthogonal factors, a fixed spectrum, a particular optimization chart, and the numerical policy of the implementation.

Table 37: Budget sweep for the attention-only controlled POET variant.

Tokens	POET	OrthAdam $d = 1$	AdamW	POET–AdamW
0.59B	3.8418	4.2115	4.2121	−0.3703
2.0B	3.1019	3.1130	3.1625	−0.0605
6.0B	2.8430	2.8023	2.8350	+0.0080
30.1B	2.6660	2.6207	2.6471	+0.0188

[2] POET: `https://github.com/Sphere-AI-Lab/poet/`.

Table 38: POET's released harness at 130M and 197M tokens, averaged over 3 seeds per cell. Each entry is the best final loss of a learning-rate sweep run separately for each method and precision.

Storage	AdamW	POET	POET–AdamW
Full bfloat16 (released)	3.9427	3.9115	-0.0312
fp32	3.7933	3.7846	-0.0087

The released harness casts the full model, including AdamW parameters and moments, to bfloat16. Standard mixed precision (fp32 parameters and moments, bfloat16 compute) gives AdamW 3.7854; the released POET kernels do not run under this mixed-precision path without modification. The longer-budget table uses a separate attention-only, full-width implementation. Identity-chart initialization improves its long-budget arm by 0.0111; merge/reinitialize changes it by 0.0005; making the spectrum trainable improves it by 0.0068. These ablations separate precision, initialization, and spectrum flexibility.

Q Pion Comparison Details

We reprodce Pion using the official code [3]. Pion reports final validation loss 2.7350 against 2.7700 for AdamW for a 1.3B model trained on 54B tokens at context length 256, in bfloat16 with the learning rate 0.0005 selected for AdamW (Shi et al., 2026). The released configuration sets AdamW parameters and ordinary momentum to bfloat16, whereas Pion's Lie-algebra momentum is created in fp32. We rerun the comparison with the released code at 1.3B, context length 4096, and $2\times$ the Chinchilla budget (48,584 steps, 50.9B C4 tokens, global batch 256), evaluating on C4-val. Every arm uses main learning rate 0.0005 with cosine decay to 0.000005, $\beta = (0.9, 0.95)$, and weight decay 0.1; AdamW uses 500 warmup steps. The paper configuration of Pion uses $\beta_2 = 0.95$, alternating left and right updates, and a second moment; the repository default uses $\beta_2 = 0.999$, updates both sides at every step, and omits the second moment. The Pion learning rate applies to the orthogonal updates only.

Table 39: Pion at 1.3B, context length 4096, $2\times$ Chinchilla. Differences are relative to bfloat16 AdamW.

Arm	Final loss	Δ vs. bfloat16 AdamW
AdamW, fp32	**2.2655**	-0.0804
AdamW, bfloat16 (released)	2.3458	0
Pion, paper configuration, Pion LR 0.00025	2.5786	$+0.2327$
Pion, paper configuration, Pion LR 0.0005	2.4175	$+0.0717$
Pion, paper configuration, Pion LR 0.001	2.3609	$+0.0150$
Pion, repository configuration, Pion LR 0.0005	2.4312	$+0.0854$

The paper configuration improves on the repository configuration by 0.0136 at the same learning rate. The best Pion learning rate, 0.001, is the largest in the completed grid; it trails 0.0005 until step 29,100 and finishes 0.0567 lower, so short learning-rate screens misrank the two. Even so, Pion remains 0.0150 behind bfloat16 AdamW and 0.0954 behind fp32 AdamW.

R StelLA Detailed Results

Table 40: StelLA controls at $S_0 = I$ over four paired seeds. Intervals are paired 95% t-intervals for the difference from Sphere.

Method	Accuracy (mean $\pm$ SD)	Δ vs. Sphere
Euclidean	84.537 ± 0.197	$-2.116\ [-2.500, -1.732]$
Euclidean + factor decay	86.408 ± 0.041	$-0.245\ [-0.589, 0.098]$
Product of spheres	$\mathbf{86.653} \pm 0.190$	0
Canonical Stiefel	86.564 ± 0.159	$-0.089\ [-0.187, 0.009]$

[3] Pion: `https://github.com/Sphere-AI-Lab/pion`.

Adaptation protocol. We reprodce StelLA using the official code [4]. All runs use StelLA's default $S_0 = I$. The checkpoint selector uses token-level loss on a fixed 120-example validation split; scores are macro-averaged over the same eight commonsense tasks as in the main text.

Table 41: Per-seed StelLA controls at $S_0 = I$. *The direct-polar update is a single seed update-rule check.

Method	Seed 42	Seed 43	Seed 44	Seed 45	Mean ± SD
Euclidean	84.405	84.814	84.387	84.543	84.537 ± 0.197
Euclidean + U/V decay (1.0)	86.468	86.399	86.385	86.380	86.408 ± 0.041
Product of spheres	86.514	86.616	86.553	86.930	86.653 ± 0.190
Canonical Stiefel	86.508	86.536	86.422	86.791	86.564 ± 0.159
Direct-polar Stiefel*	86.641	–	–	–	–

The paired Sphere–Euclidean difference is 2.116 accuracy points (95% t-interval $[1.732, 2.500]$). The Stiefel–Sphere difference is -0.089 points ($[-0.187, 0.009]$), with Stiefel below Sphere in all four seeds, and the factor-decay–Sphere difference is -0.245 points ($[-0.589, 0.098]$).

Direct-polar Stiefel omits only the post-Adam tangent projection and scores 86.641, compared with 86.508 for canonical Stiefel in the same seed-42 check. The following sensitivity tables come from a separate seed-42 protocol replication and are not pooled with the paired main-table estimates.

Table 42: Learning-rate sensitivity, scored by eight-task macro accuracy. These scores are not used to select the main-table arms.

Method	10^{-4}	5×10^{-4}	10^{-3}
Euclidean, no decay	84.602	84.596	**84.777**
Product of spheres	84.835	**86.650**	86.136
Canonical Stiefel	84.835	**86.690**	86.256

Sphere and Stiefel attain their best tested value at the default 5×10^{-4}. Euclidean is best at the upper grid edge 10^{-3} and remains 1.87 points below Sphere.

Table 43: Seed-42 factor-wise U/V-decay sensitivity at learning rate 5×10^{-4}, scored by eight-task macro accuracy. The middle matrix S has zero decay.

U/V weight decay	Accuracy
0	84.596
0.01	84.542
0.03	85.187
0.1	85.148
0.3	86.250
1.0	86.732

The largest tested U/V decay, 1.0, comes within 0.25 points of Sphere in the four-seed comparison (95% paired interval $[-0.59, 0.10]$), recovering about 88% of the Euclidean-to-Sphere gain. The learning-rate and decay tables are post-hoc sensitivity checks; main-table checkpoints are selected by loss on the fixed 120-example validation split.

S PSOFT Detailed Results

Protocol. We reprodce PSOFT using the official code [5]. All arms adapt Llama-3.1-8B on Commonsense-15K with adapters on the query and value matrices, rank 424, and three epochs, and they share PSOFT's selected principal subspace, effective initialization, and learned magnitude vectors α, β (Wu et al., 2026). They differ only in the latent 424×424 transformation R: PSOFT's five-term Cayley/Neumann transformation, an unconstrained dense R, or a dense R whose rows are initialized at unit norm and kept on unit spheres by tangent projection of the AdamW update followed

[4] StelLA: `https://github.com/SonyResearch/stella`.

[5] StelLA: `https://github.com/fei407/PSOFT`.

by row renormalization. Scores are equal-weight macro accuracy over the same eight commonsense tasks as in the StelLA study. Learning rates are selected by the mean best validation loss over seeds 42 and 43: 8×10^{-4} for PSOFT and 4×10^{-4} for row-sphere. The dense arm uses 4×10^{-4}, the validation-best rate of a seed-42 sweep.

Table 44: PSOFT controls by learning rate and seed, scored by eight-task macro accuracy. Bold marks the validation-selected rate of each arm.

Latent transformation	LR	Seed 42	Seed 43	Seed 44	Mean
PSOFT (Cayley/Neumann)	4×10^{-4}	81.371	81.488	80.933	81.264
	$\mathbf{8 \times 10^{-4}}$	80.503	80.444	80.694	80.547
Dense	$\mathbf{4 \times 10^{-4}}$	81.047	80.984	80.828	80.953
	8×10^{-4}	81.439	81.237	80.936	81.204
Row-sphere	$\mathbf{4 \times 10^{-4}}$	81.035	81.287	80.891	81.071
	8×10^{-4}	81.264	81.475	80.829	81.189

At the selected rates, row-sphere exceeds PSOFT by 0.52 points, higher in all three paired seeds, and exceeds the dense arm by 0.12 points, with per-seed differences -0.01, $+0.30$, and $+0.06$. At a shared 8×10^{-4}, the dense arm matches row-sphere (81.20 versus 81.19), so the sphere constraint gives no measurable gain there. At a shared learning rate the ordering of PSOFT and row-sphere reverses: PSOFT leads by 0.19 points at 4×10^{-4} and trails by 0.64 at 8×10^{-4}, in each case in all three seeds. The comparison therefore supports no accuracy cost from replacing the orthogonal transformation with independently normalized rows, but not a learning-rate-independent advantage in either direction.

Geometry. We inspect the final seed-42 adapters at 4×10^{-4}, 64 matrices of size 424×424 per arm. All 27,136 row-sphere rows have unit norm to within 1.2×10^{-7}. Dense row norms drift from their unit initialization by 0.006 on average and at most 0.165. Reconstructing PSOFT's five-term Neumann transformation gives $\|RR^\top - I\|_F/\sqrt{424}$ of 0.010 on average and at most 0.025, with singular values between 0.864 and 1.000. The implemented transformation is therefore approximately rather than exactly orthogonal.

T LIMITATIONS

This work focuses exclusively on hard orthogonality constraints, rather than soft constraints imposed through a regularization term such as $\|WW^\top - I\|_F$. Soft constraints introduce an additional regularization coefficient that must be tuned, whereas hard constraints fix the weight norms and thereby eliminate the need for weight decay. Reducing the number of hyperparameters may simplify the training and tuning of large language models.

This work tunes only the two hyperparameters that most strongly affect performance: learning rate and weight decay, using grids with values spaced by factors of 2 or 3. Although the optimal values may lie between the tested candidates, an optimizer that fails to outperform competing methods under such a grid search is unlikely to offer improvements that generalize across different model and dataset scales. The AdamW learning rate for embedding/output-head parameters, normalization weights, and gains is shared rather than tuned across the Muon-family arms, so comparisons within the ladder are matched, whereas absolute gaps to Muon and MuonH are conditional on this setting. The pretraining study uses fixed contiguous row blocks and reports validation loss rather than downstream evaluations; module-specific assignments are tested only at 360M, and learned groupings define complementary interventions (Yoshihara, 2026). The adaptation studies cover 8B-class Llama models on commonsense reasoning.