Symmetry-Compatible Principle for Optimizer Design: Embeddings, LM Heads, SwiGLU MLPs, and MoE Routers*

Tim Tsz-Kit Lau[†‡] **Weijie Su**[†]
[†]University of Pennsylvania, Philadelphia, PA 19104
[‡]DRW, Palo Alto, CA 94301
timlautk@gmail.com suw@wharton.upenn.edu

Abstract

A striking geometric disparity has long persisted in the practice of deep learning. While modern neural network architectures naturally exhibit rich symmetry and equivariance properties, popular optimization methods, such as ADAM and its variants, operate inherently coordinate-wise, rendering them unable to respect the equivariance structures of the parameter space. In this paper, we address this disparity by introducing a *symmetry-compatible principle* for optimizer design. Specifically, we argue that the gradient update rule should be equivariant under the symmetry group acting on the corresponding weight block, and should remove symmetry-redundant directions when the parameterization has quotient structure. Following this principle, we first provide a unified perspective on bi-orthogonally equivariant updates for general matrix layers, as employed by stochastic spectral descent, MUON, SCION, and polar gradient methods. More importantly, by moving from orthogonal groups to permutation and shared-shift symmetries, we derive new classes of symmetry-compatible optimizers tailored to parameter blocks whose symmetries differ from those of ordinary matrix layers: for embedding and LM head matrices, left-permutation and right-orthogonal equivariance leads to one-sided spectral, row-norm, and hybrid row-norm/spectral updates, with projected variants for LM heads; for SwiGLU MLP projections, intermediate-neuron permutation symmetry motivates row-aware and column-aware variants; and for MoE routers, expert-permutation symmetry together with shared-logit-shift invariance gives rise to projected centered row-norm and left-spectral updates. These constructions yield an end-to-end layerwise optimizer stack in which each major matrix-valued parameter class is assigned an update whose equivariance matches its symmetry group. We corroborate this optimizer design principle through extensive pre-training experiments on dense and sparse MoE language models, including Qwen3-0.6B-style, Gemma 3 1B-style, OLMoE-1B-7B-style, and downsized gpt-oss architectures. Across these experiments, symmetry-compatible update rules consistently improve final validation loss, reduce expert load imbalance in sparse MoE models, and in several cases control final vocabulary-logit growth, improve router stability, and overall training stability over the corresponding ADAMW updates.

https://github.com/timlautk/equivariant_optimizers

1 Introduction

The most widely used optimizers in deep learning, such as ADAM [77], ADAFACTOR [128], RMSPROP [141], ADAGRAD [39, 106], and their variants, all belong to the broad family of *coordinate-*

*See arXiv:2605.18106 for a detailed version of this work with extended results.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

wise adaptive gradient methods. These methods treat model parameters as a single long concatenated vector and update each coordinate independently. Despite their empirical success, this design implicitly assumes that every entry of a weight matrix is an independent coordinate in a high-dimensional vector space. This assumption is rarely questioned, yet it strongly shapes the training dynamics of modern neural networks. In particular, such a geometry-blind treatment ignores the rich matrix structure of neural network parameters and fails to distinguish between the geometries of different layer types, such as embeddings, LM heads, dense linear layers, attention projections, SwiGLU MLP projections, and MoE routers.

At the same time, our theoretical understanding of optimizer behavior remains limited across the two major families most relevant to modern large-scale training: *coordinate-wise adaptive gradient optimizers* and *spectral optimizers*. In language model pre-training in particular, comparisons between these optimizer families are still largely empirical, relying on large-scale benchmarking exercises [126, 145] and speedrunning [72], with relatively little analysis of their different geometric behavior and training dynamics. Hyperparameter transfer rules [155] and scaling-law prescriptions [64, 74], for example, are often applied across optimizers, even though their original development was tied primarily to coordinate-wise adaptive methods, particularly AdamW [101]. Such benchmarking practices implicitly assume that these optimizer families are directly comparable and share similar training phenomena, which need not be the case.

Contributions. Our work makes the following contributions.

1. **A symmetry-compatible principle for matrix-gradient optimizer design.** We argue that optimizer updates should respect the symmetry structure of the parameter block on which they act. Popular coordinate-wise adaptive optimizers such as Adam, AdamW, and RMSprop are often *geometrically mismatched* for matrix-valued parameters, since their updates generally fail to respect the natural equivariance and quotient structures of modern neural network layers. Fully-connected layers, attention projections, embedding and LM head matrices, SwiGLU MLP projections, and MoE router matrices all possess nontrivial row, column, permutation, spectral, or shared-shift geometries. Our central message is that neural network weight matrices live in layer-dependent geometries that coordinate-wise adaptive methods do not explicitly capture.
2. **A unifying equivariance view of spectral optimizers.** We show that optimizer updates governed by orthogonal equivariance naturally lead to a broad class of *spectral optimizers*. This class includes stochastic spectral descent (SSD) [20], Muon [73], Scion [113], and polar gradient methods (PolarGrad) [83]. These methods compute, exactly or approximately, the orthogonal polar factor of an update direction D, such as a gradient G or momentum M: $D = U\Sigma V^\top \Rightarrow U_{\mathsf{p}} := \mathrm{polar}(D) = UV^\top$. Such updates are bi-orthogonally equivariant, preserve the singular-vector geometry of the update direction, and arise naturally from matrix geometry. This perspective also gives a symmetry-based interpretation of the spectral-norm steepest descent principle underlying Muon [12, 13, 73]: because the spectral norm is unitarily invariant, the corresponding polar update is naturally bi-orthogonally equivariant.
3. **Layerwise equivariant optimizers for architecture–optimizer co-design.** Beyond full spectral optimizers for ordinary matrix layers, we derive symmetry-compatible optimizer classes for layers whose symmetries differ from those of standard linear maps. For embeddings and LM heads, left-permutation/right-orthogonal equivariance leads to one-sided spectral, row-norm, and hybrid row-norm/spectral updates; for LM heads, softmax shared-logit-shift invariance further requires projected updates on the horizontal quotient space. For SwiGLU MLP projections, intermediate-neuron permutation symmetry motivates row-aware updates for gate and up projections and column-aware updates for down projections. For MoE routers, expert-permutation symmetry together with shared-logit-shift invariance leads to projected centered row-norm, left-spectral, and hybrid updates. The corresponding practical momentum variants are denoted RightPolarGradM, LeftPolarGradM, RowNormM, and HybridPolarGradM. These constructions instantiate an architecture–optimizer co-design principle based on layerwise equivariance and quotient geometry.
4. **End-to-end pre-training evidence.** We evaluate the proposed layerwise optimizer assignments in dense and sparse MoE language model pre-training experiments (Sections 5 and K). To the best of our knowledge, these experiments provide the first end-to-end pre-training optimizer stack in which all major matrix-valued parameter classes in language models are assigned updates according to their layerwise symmetry. Replacing AdamW on large vocabulary-indexed matrices

with row-norm or hybrid symmetry-compatible updates consistently improves final validation loss. The gains are modest but visible for the smaller Qwen3-0.6B-style dense model, become more pronounced for the larger Gemma 3 1B-style model (Figure 1), and persist in sparse MoE experiments based on OLMoE-1B-7B and downsized gpt-oss. In dense models, hybrid row-norm/spectral updates for SwiGLU MLP projections further improve validation loss, and projected symmetry-compatible LM head updates reduce final vocabulary-logit growth in Gemma 3 1B-style pre-training. In the MoE setting, symmetry-compatible router updates improve over coordinate-wise router updates, reduce expert load imbalance, improve router stability as measured by router z-loss, and can reduce training loss spikes.

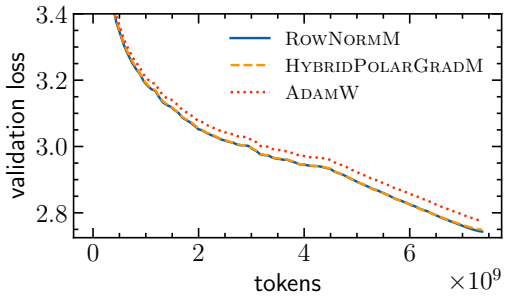


Figure 1: Validation losses for Gemma 3 1B-style pre-training. The configurations differ only in the optimizer assigned to the embedding and LM head matrices.

Organization. In Section 2, we develop the layerwise symmetry-compatible principle, beginning from a linear-operator view of matrix parameters and the resulting coordinate-free equivariance requirements. We then derive a general class of matrix-aware optimizers for deep learning in Section 3, coined *spectral optimizers*, which characterize all matrix-gradient optimizers with bi-orthogonally equivariant updates. Then, making use of the principle of symmetry and equivariance, we derive a wider class of symmetry-compatible optimizers in Section 4. This class includes one-sided variants of spectral optimizers, i.e., left- and right-spectral optimizers, as well as row-norm-based and hybrid variants, which are especially relevant for non-standard matrix layers such as embeddings, LM heads, SwiGLU MLPs and MoE routers. To demonstrate the empirical implications of our proposed symmetry-aware matrix view and symmetry-compatible optimizers, we perform pre-training experiments on both dense and MoE language models in Sections 5 and K. We end the paper with some concluding remarks in Section 6.

2 Equivariant Optimizers from Layerwise Symmetry

Modern deep learning architectures contain matrix-valued parameters with different symmetry structures. The common principle is that a parameter matrix does not always represent an arbitrary array of coordinates, but often represents a linear map between two structured spaces. If the coordinates of these spaces are changed, the parameter and its gradient transform accordingly, and a geometry-compatible optimizer should transform in the same way.

We first state this principle in a general form. Let $W \in \mathbb{R}^{m\times n}$ represent a linear map from an input space to an output space. Suppose the output and input coordinates are transformed by invertible matrices $P \in \mathrm{GL}(m)$ and $Q \in \mathrm{GL}(n)$. Then the same linear map is represented by $\widetilde{W} = PWQ^{-1}$. If $\widetilde{f}(\widetilde{W}) \coloneqq f(P^{-1}\widetilde{W}Q)$, then standard matrix calculus gives

$$\nabla_{\widetilde{W}} \widetilde{f}(\widetilde{W}) = P^{-\top} \nabla_W f(W) Q^{\top}.$$

Thus, under a general change of coordinates, the gradient transforms contravariantly with respect to the output coordinates and covariantly with respect to the input coordinates.

In this work, we study the equivariance of the update map $\mathcal{U}\colon \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ in matrix-optimizer iterations

$$(\forall k \in \mathbb{N}) \qquad W_{k+1} = W_k - \gamma_k \mathcal{U}(D_k),$$

where D_k is an update direction, such as a gradient or momentum. The relevant requirement is not necessarily that the layerwise loss function be invariant under arbitrary transformations, but that the optimizer update transform consistently with the representation of its input direction. Thus, once a layer symmetry specifies a transformation law $D_k \mapsto g \cdot D_k$, we require

$$\mathcal{U}(g \cdot D_k) = g \cdot \mathcal{U}(D_k).$$

When D_k transforms equivariantly, the update $\mathcal{U}(D_k)$ therefore transforms equivariantly as well.

In this paper, however, we do not require equivariance under all invertible changes of coordinates. The relevant symmetry group depends on the layer. For ordinary linear and attention matrices, the natural coordinate changes are orthonormal changes of basis, so $P \in \mathbb{O}^m$ and $Q \in \mathbb{O}^n$. In this case $P^{-\top} = P$ and $Q^{-1} = Q^\top$, and both the parameter and gradient transform as $W \mapsto PWQ^\top$ and $G \mapsto PGQ^\top$. This leads to the bi-orthogonal equivariance condition

$$\mathcal{U}(PGQ^\top) = P\,\mathcal{U}(G)\,Q^\top.$$

For embedding and LM head matrices $W \in \mathbb{R}^{v\times d}$, the row axis indexes vocabulary items, so the admissible left action is not a general orthogonal rotation but a permutation $P \in \mathbb{P}^v$, while the hidden feature axis still admits right orthogonal transformations. For MoE routers, the row axis indexes experts and additionally has a shared-logit-shift invariance. For SwiGLU MLP projections, the relevant symmetry is permutation of intermediate neurons, which acts on the rows of the gate and up projections and on the columns of the down projection.

This gives a layerwise equivariance principle: the optimizer update map should commute with the symmetry group of the parameter block on which it acts. Full bi-orthogonal equivariance leads to spectral optimizers for ordinary matrix layers; left-permutation/right-orthogonal equivariance leads to row-aware and right-spectral optimizers for embeddings and LM heads; intermediate-neuron permutation symmetry leads to row- and column-aware updates for SwiGLU MLP projections; and expert-permutation plus shared-shift symmetry leads to centered row-aware or left-spectral updates for MoE routers.

2.1 A General Symmetry-Induced Optimizer Geometry

Let $W \in \mathbb{R}^{m\times n}$ be a layer parameter and let $f\colon \mathbb{R}^{m\times n} \to \mathbb{R}$ be the corresponding layerwise loss. Suppose a group $\mathcal{G}$ acts on the parameter space by transformations $W \mapsto g \cdot W$. In the matrix settings considered below, this action is typically of the form $g \cdot W = PWQ^{-1}$, or, after restricting to orthogonal or permutation symmetries, $g \cdot W = PWQ^\top$. We say that the parameterization admits the symmetry group $\mathcal{G}$ if $f(g \cdot W) = f(W)$ for all $g \in \mathcal{G}$. The corresponding optimizer update map should satisfy

$$(\forall g \in \mathcal{G}) \qquad \mathcal{U}(g \cdot G) = g \cdot \mathcal{U}(G),$$

where G is an update direction, such as a gradient or momentum, expressed in the corresponding transformed coordinates. This condition ensures that the optimizer does not depend on arbitrary choices of representation that are invisible to the model.

2.2 Bi-Orthogonal Equivariance for Ordinary Matrix Layers

The general reparameterization $W \mapsto PWQ^{-1}$ specializes to $W \mapsto PWQ^\top$ when the admissible coordinate changes are orthogonal. This is the natural case for ordinary linear layers and attention projection matrices, where both input and output coordinates represent continuous feature bases. Therefore an update map for such layers should satisfy

$$(\forall P \in \mathbb{O}^m)(\forall Q \in \mathbb{O}^n) \qquad \mathcal{U}(PGQ^\top) = P\,\mathcal{U}(G)Q^\top.$$

This is exactly *bi-orthogonal equivariance*.

Definition 2.1 (Bi-orthogonal equivariance)**.** Let $\mathcal{U}\colon \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ be a matrix-valued map. We say that $\mathcal{U}$ is *bi-orthogonally equivariant* if, for all $G \in \mathbb{R}^{m\times n}$ and all $P \in \mathbb{O}^m$, $Q \in \mathbb{O}^n$,

$$\mathcal{U}(PGQ^\top) = P\,\mathcal{U}(G)Q^\top.$$

Thus, bi-orthogonal equivariance is exactly the requirement that if W and its gradient are transformed as $W \mapsto PWQ^\top$ and $G \mapsto PGQ^\top$, then the optimizer update transforms in the same way. More generally, if an update rule has the form $\Delta W = \mathcal{U}(W, G)$, one may require

$$\mathcal{U}(PWQ^\top, PGQ^\top) = P\,\mathcal{U}(W, G)Q^\top.$$

In this paper, we focus on update maps of the form $\mathcal{U}(G)$ or $\mathcal{U}(D)$, where D is a gradient-derived direction such as momentum. Terms that depend explicitly on W, such as decoupled weight decay or scalar step-size scaling, can typically be handled separately and preserve the same equivariance, so we suppress the dependence on W for simplicity.

For ordinary matrix layers, bi-orthogonal equivariance motivates polar and spectral update directions. In particular, the orthogonal polar factor [9, 61] satisfies

$$\operatorname{polar}(PGQ^\top) = P \operatorname{polar}(G) Q^\top, \tag{1}$$

and hence MUON-style and polar-gradient updates are symmetry-compatible for ordinary matrix layers. We defer the full characterization of bi-orthogonally equivariant maps as spectral operators to Section 3. Standard momentum constructions such as EMA, Polyak, and Nesterov momentum preserve the same equivariance because their buffers are linear combinations of past gradients. We state this fact for Nesterov momentum in the following proposition.

Proposition 2.1 (Nesterov momentum is bi-orthogonally equivariant)**.** *Let $(W_k)_{k\in\mathbb{N}} \subset \mathbb{R}^{m\times n}$ be a parameter sequence, and define $G_k := \nabla_W f(W_k)$ for $k \in \mathbb{N}$. Fix orthogonal matrices $P \in \mathbb{O}^m$ and $Q \in \mathbb{O}^n$, and define the transformed parameter sequence $\widetilde{W}_k := PW_kQ^\top$ for $k \in \mathbb{N}$, with transformed gradients $\widetilde{G}_k := \nabla_W f(\widetilde{W}_k)$ for $k \in \mathbb{N}$. Let the momentum buffer be $M_k = \beta M_{k-1} + G_k$ with $M_{-1} = 0$, and define the update direction by $N_k := G_k + \beta M_k$. For the transformed sequence, let $\widetilde{M}_k = \beta \widetilde{M}_{k-1} + \widetilde{G}_k$ with $\widetilde{M}_{-1} = 0$, and $\widetilde{N}_k := \widetilde{G}_k + \beta \widetilde{M}_k$. Then we have $\widetilde{M}_k = PM_kQ^\top$, $\widetilde{N}_k = PN_kQ^\top$, and $\operatorname{polar}(\widetilde{N}_k) = P \operatorname{polar}(N_k) Q^\top$.*

3 Spectral Optimizers as the Bi-Orthogonally Equivariant Class

We have used bi-orthogonal equivariance as the symmetry principle for ordinary matrix layers. We now record the corresponding structural characterization: direction-wise update maps satisfying this equivariance are precisely spectral operators. This explains why SSD [22], MUON [73], SCION [113], POLARGRAD [83], and related spectral updates form the canonical optimizer class for ordinary matrix layers. The preceding sections show how different symmetry groups lead instead to different optimizer classes for embeddings, LM heads, SwiGLU MLP projections, and MOE routers.

We first recall the relevant matrix-analytic notions. A proper function $\varphi\colon \mathbb{R}^{m\times n} \to \overline{\mathbb{R}}$ is called *spectral* if there exists a proper function $\psi\colon \mathbb{R}^r \to \overline{\mathbb{R}}$, where $r = \min\{m, n\}$, such that $\varphi = \psi \circ \sigma$. Its matrix-valued analogue is a *spectral operator*: a map $\mathfrak{U}\colon \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ is spectral if there exists an absolutely symmetric map $\psi\colon \mathbb{R}^r \to \mathbb{R}^r$ such that, for every singular value decomposition $D = U \operatorname{Diag}(\sigma(D)) V^\top$,

$$\mathfrak{U}(D) = U \operatorname{Diag}(\psi(\sigma(D))) V^\top.$$

Thus, spectral operators preserve the singular-vector geometry of their input and act only on singular values.

We next recall the standard gradient formula for spectral functions; see, e.g., Lewis [85].

Theorem 3.1 (Gradient formula for spectral functions)**.** *Let $f\colon \mathbb{R}^r \to \overline{\mathbb{R}}$ be convex and absolutely symmetric. Then the corresponding spectral function $f \circ \sigma$ is differentiable at $W \in \mathbb{R}^{m\times n}$ if and only if f is differentiable at $\sigma(W)$. In this case, if $W = U \operatorname{Diag}(\sigma(W)) V^\top$ is a singular value decomposition of W, then $\nabla(f \circ \sigma)(W) = U \operatorname{Diag}(\nabla f(\sigma(W))) V^\top$.*

This formula shows that spectral scalar functions act through singular values while preserving singular directions. The same structure appears when one requires an update map to commute with arbitrary left and right orthogonal changes of coordinates. We now state the corresponding characterization.

Theorem 3.2 (Characterization of bi-orthogonally equivariant update maps)**.** *A continuous matrix-valued map $\mathfrak{U}\colon \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ satisfies $\mathfrak{U}(PDQ^\top) = P\,\mathfrak{U}(D)\,Q^\top$ for all $D \in \mathbb{R}^{m\times n}$, $P \in \mathbb{O}^m$, and $Q \in \mathbb{O}^n$ if and only if it is a spectral operator. Equivalently, $\mathfrak{U}$ is bi-orthogonally equivariant if and only if, for every singular value decomposition $D = U \operatorname{Diag}(\sigma(D)) V^\top$, there exists an absolutely symmetric map $\psi\colon \mathbb{R}^r \to \mathbb{R}^r$ such that $\mathfrak{U}(D) = U \operatorname{Diag}(\psi(\sigma(D))) V^\top$, where $r = \min\{m, n\}$.*

We denote the set of bi-orthogonally equivariant matrix maps from $\mathbb{R}^{m\times n}$ to $\mathbb{R}^{m\times n}$ by $\mathfrak{U}_{\mathbb{O}}^{m\times n}$. Theorem 3.2 shows that bi-orthogonal equivariance is not merely a desirable property: it is a complete structural characterization of direction-wise matrix update maps. Any such update must act through the singular values of the update direction and preserve its singular vectors.

Remark 3.1. Theorem 3.2 concerns equivariance of the optimizer update map with respect to transformations of the matrix gradient or update direction. It does not assume that the layerwise

loss itself satisfies $f(PWQ^\top) = f(W)$ for every individual layer parameter W. The claim is instead that, whenever a matrix gradient or momentum direction is represented in a rotated basis, a symmetry-compatible optimizer should transform its update in the same way.

Accordingly, we call a matrix optimizer *spectral* if its update rule is bi-orthogonally equivariant. Concretely, its iterates satisfy

$$W_{k+1} = W_k - \gamma_k \mathcal{U}(D_k),$$

where $\gamma_k > 0$, $\mathcal{U} \in \mathcal{U}_{\mathbb{O}}^{m\times n}$, and D_k is an update direction that itself transforms bi-orthogonally, such as the gradient or a symmetry-compatible momentum direction. By Theorem 3.2, every such direction-wise spectral optimizer is determined by a singular-value transformation ψ.

This characterization applies to maps acting on a single update direction. It does not exclude broader stateful matrix optimizers such as SHAMPOO, whose auxiliary states may themselves evolve equivariantly. Spectral optimizers should therefore be understood as the bi-orthogonally equivariant class of memoryless, or direction-wise, matrix update maps; stateful equivariant optimizers form a larger class.

4 General Symmetry-Compatible Optimizers via Equivariance

In modern deep learning architectures, different matrix-valued parameters exhibit different underlying symmetry structures. For standard linear and attention layers, both the input and output spaces are continuous and admit arbitrary changes of orthonormal bases, leading naturally to bi-orthogonal equivariance. By contrast, embedding matrices and LM heads possess a discrete symmetry along their vocabulary dimension, namely permutation equivariance. Such permutation-based symmetries are largely specific to deep learning architectures and rarely arise in classical matrix optimization problems from other areas of machine learning, statistics, and data science. In the following, we develop a class of optimizers for embedding matrices and LM heads that respect the natural symmetry structure of these parameters, namely *left-permutation and right-orthogonal (LPRO) equivariance*.

4.1 Optimizers for Embeddings and LM Heads via LPRO Equivariance

For vocabulary-indexed matrices such as input embeddings and untied LM heads, rows index vocabulary items and columns correspond to hidden features. This motivates left-permutation and right-orthogonal equivariant update maps. For an update direction $D \in \mathbb{R}^{v\times d}$, representative examples include row-norm updates, right-spectral updates, and hybrid row-norm/right-spectral updates. In the embedding case, these take the form

$$\mathcal{U}_{\mathsf{row}}(D) = \mathrm{Diag}(\eta(\|D_{1:}\|_2), \dots, \eta(\|D_{v:}\|_2))D,$$
$$\mathcal{U}_{\mathsf{R}}(D) = D(D^\top D + \varepsilon I)^{-1/2},$$

and

$$\mathcal{U}_{\mathsf{hyb}}(D) = \mathcal{U}_{\mathsf{R}}(\mathcal{U}_{\mathsf{row}}(D)) \quad \text{or} \quad \mathcal{U}_{\mathsf{row}}(\mathcal{U}_{\mathsf{R}}(D)).$$

The row-norm update acts locally on vocabulary rows, the right-spectral update acts globally through the hidden-feature Gram matrix, and the hybrid update combines the two.

For untied LM heads, there is an additional quotient geometry. Since softmax probabilities are invariant under adding the same scalar to all vocabulary logits, adding a shared row vector $\mathbf{1}_v a^\top$ to the LM head changes $W_{\mathrm{out}}h$ only by a shared logit shift and therefore leaves the output distribution unchanged. Thus, the shared-row direction is vertical, and the intrinsic LM head update should be horizontal. Let

$$\Pi_v^\perp := I_v - \frac{1}{v}\mathbf{1}_v\mathbf{1}_v^\top.$$

For LM head updates, we therefore apply the corresponding row-norm, right-spectral, or hybrid maps to the centered direction $D_c = \Pi_v^\perp D$, and project the final update back to the horizontal subspace when rowwise nonlinear operations are used. For example, the horizontal LM head row-norm update is

$$\mathcal{U}_{\mathsf{row}}^{\mathrm{LM}}(D) = \Pi_v^\perp \,\mathrm{Diag}(\eta(\|D_{c,1:}\|_2), \dots, \eta(\|D_{c,v:}\|_2))D_c, \qquad D_c = \Pi_v^\perp D.$$

The final projection is not redundant: although $\mathbf{1}_v^\top D_c = 0$, the rowwise-rescaled matrix

$$\mathrm{Diag}(\eta(\|D_{c,1:}\|_2), \dots, \eta(\|D_{c,v:}\|_2))D_c$$

need not be row-centered, since the scaling factors vary across rows. Similarly, for hybrid row-norm/right-spectral LM head updates, we reproject after rowwise nonlinear stages and, in practice, also project the final update to ensure that the actual parameter update remains horizontal.

We now explain why these updates are natural from the symmetry of embeddings and LM heads. In empirical uses of MUON [73], it is often recommended that ADAMW be used for embedding and LM head matrices. For embeddings, this recommendation is motivated in part by modular norm theory [81]; for LM heads, it appears to be driven more by empirical considerations. Relatedly, SCION [113] derives embedding updates from induced operator norms in its linear minimization oracle framework. These approaches depend on a particular choice of norm. Here we instead derive optimizer classes directly from the symmetry of the parameterization.

Let $v \in \mathbb{N}^*$ denote the vocabulary size and $d \in \mathbb{N}^*$ the embedding dimension, typically with $v \gg d$. Consider an input embedding matrix $E \in \mathbb{R}^{v\times d}$ and an untied LM head matrix $W_{\text{out}} \in \mathbb{R}^{v\times d}$. In both cases, rows index vocabulary items, while columns correspond to hidden features. Thus, the row axis admits permutation symmetry, whereas the hidden feature axis admits right orthogonal symmetry. The natural equivariance condition for an update map is therefore

$$\mathcal{U}_{\mathsf{LPRO}}(PDR^\top) = P\,\mathcal{U}_{\mathsf{LPRO}}(D)R^\top, \tag{2}$$

for all $D \in \mathbb{R}^{v\times d}$, permutation matrices $P \in \mathbb{P}^v$, and orthogonal matrices $R \in \mathbb{O}^d$. We call such maps *left-permutation right-orthogonal* (LPRO) equivariant. For untied LM heads, the same principle is applied on the horizontal quotient space defined by $\Pi_v^\perp$.

Definition 4.1 (Left-permutation and right-orthogonal equivariant maps)**.** A map $\mathcal{U}_{\mathsf{LPRO}}\colon \mathbb{R}^{v\times d} \to \mathbb{R}^{v\times d}$ is said to be *left-permutation and right-orthogonal equivariant* if (2) holds for all $D \in \mathbb{R}^{v\times d}$, $P \in \mathbb{P}^v$, and $R \in \mathbb{O}^d$. We denote the set of such maps by $\mathcal{U}_{\mathsf{LPRO}}^{v\times d}$.

4.1.1 Right-Spectral Optimizers

A first natural subclass of LPRO-equivariant maps is given by right-spectral updates,

$$\mathcal{U}_{\mathsf{R}}(D) = D\,\Phi(D^\top D), \tag{3}$$

where $\Phi\colon \mathbb{S}_+^d \to \mathbb{R}^{d\times d}$ is an orthogonally equivariant spectral operator. Equivalently, if $D^\top D = V\,\mathrm{Diag}(\lambda(D^\top D))V^\top$, then

$$\Phi(D^\top D) = V\,\mathrm{Diag}(\psi(\lambda(D^\top D)))V^\top$$

for some absolutely symmetric map $\psi\colon \mathbb{R}_+^d \to \mathbb{R}^d$.

Theorem 4.1 (Right-spectral updates are LPRO-equivariant)**.** *Let $\mathcal{U}_{\mathsf{R}}$ be of the form (3), where $\Phi(RXR^\top) = R\Phi(X)R^\top$ for all $X \in \mathbb{S}_+^d$ and $R \in \mathbb{O}^d$. Then*

$$\mathcal{U}_{\mathsf{R}}(PDR^\top) = P\,\mathcal{U}_{\mathsf{R}}(D)R^\top$$

for all $D \in \mathbb{R}^{v\times d}$, $P \in \mathbb{P}^v$, and $R \in \mathbb{O}^d$. We denote the set of right-spectral maps by $\mathcal{U}_{\mathsf{R}}^{v\times d}$.

The choice $\Phi(X) = (X + \varepsilon I)^{-1/2}$ yields the damped right-polar update

$$\mathcal{U}_{\mathsf{R}}(D) = D(D^\top D + \varepsilon I)^{-1/2}.$$

When $\varepsilon = 0$ and D has full column rank, this is the orthogonal polar factor of D. Thus, right-spectral updates are the one-sided analogue of spectral or polar-gradient updates, but they require only the smaller right Gram matrix $D^\top D \in \mathbb{R}^{d\times d}$.

For LM heads, right-spectral updates preserve horizontality once the input direction is centered. Indeed, if $\mathbf{1}_v^\top D_c = 0$, then $\mathbf{1}_v^\top D_c \Phi(D_c^\top D_c) = 0$. Thus, in exact arithmetic, the final projection is not necessary for purely right-spectral maps. We nevertheless use the quotient-space form

$$\mathcal{U}_{\mathsf{R}}^{\mathrm{LM}}(D) = \Pi_v^\perp D_c(D_c^\top D_c + \varepsilon I)^{-1/2}, \qquad D_c = \Pi_v^\perp D,$$

as a numerically robust horizontal update, especially when the inverse square root or polar factor is computed by an inexact oracle.

This computational distinction is important for embeddings and LM heads, where $v \gg d$. Although coordinate-wise adaptive methods such as ADAM are often used for these matrices in practice,

they are not LPRO-equivariant. The reason for their empirical use may be computational rather than geometric: accurately approximating polar factors of tall-skinny, ill-conditioned vocabulary matrices can be challenging with simple Newton–Schulz iterations. More robust polar decomposition algorithms, such as QDWH [109] and ZOLO-PD [108], can compute such updates more accurately [83].

Right-spectral normalization is also natural statistically. In a mini-batch of size b, gradients of embedding and LM head matrices have rank at most $O(b)$, since they factor through token occurrences and hidden features. Thus, even when the vocabulary dimension is very large, the gradient often lies in a low-dimensional feature subspace. Right-spectral updates act on this intrinsic singular geometry through $D^\top D$, rather than on individual coordinates.

4.1.2 Row-Norm and Hybrid LPRO-Equivariant Optimizers

Right-spectral maps do not exhaust all LPRO-equivariant updates. Restricting the left symmetry from the full orthogonal group to the permutation group allows update maps that depend on individual row norms. In particular, row-norm maps of the form

$$\mathcal{U}_{\mathsf{row}}(D) = \mathrm{Diag}(\eta(\|D_{1:}\|_2), \dots, \eta(\|D_{v:}\|_2))D \tag{4}$$

are LPRO-equivariant for any scalar function $\eta\colon \mathbb{R}_+ \to \mathbb{R}$, because left multiplication by a permutation matrix permutes the row norms and right orthogonal transformations preserve them. We denote the set of such maps by $\mathcal{U}_{\mathsf{row}}^{v\times d}$.

For LM heads, the horizontal version is

$$\mathcal{U}_{\mathsf{row}}^{\mathrm{LM}}(D) = \Pi_v^\perp \,\mathrm{Diag}(\eta(\|D_{c,1:}\|_2), \dots, \eta(\|D_{c,v:}\|_2))D_c, \qquad D_c = \Pi_v^\perp D. \tag{5}$$

This map is permutation equivariant and right-orthogonal equivariant on the horizontal quotient space, and it satisfies

$$\mathbf{1}_v^\top \mathcal{U}_{\mathsf{row}}^{\mathrm{LM}}(D) = 0.$$

The final projection in (5) is essential for row-norm updates, because row-dependent rescaling does not preserve the zero-row-mean constraint.

Thus, right-spectral maps form a global Gram-based subclass, while row-norm maps form a local row-adaptive subclass:

$$\mathcal{U}_{\mathsf{R}}^{v\times d} \subset \mathcal{U}_{\mathsf{LPRO}}^{v\times d}, \qquad \mathcal{U}_{\mathsf{row}}^{v\times d} \subset \mathcal{U}_{\mathsf{LPRO}}^{v\times d}.$$

Hybrid LPRO maps are obtained by composing these two types of maps. For LM heads, hybrid maps are obtained by the same compositions, but with horizontal projections inserted after rowwise nonlinear stages and on the final update. For example, a row-then-right-spectral horizontal hybrid update may be written as

$$D_0 = \Pi_v^\perp D, \qquad D_1 = \Pi_v^\perp \,\mathrm{Diag}(\eta(\|D_{0,1:}\|_2), \dots, \eta(\|D_{0,v:}\|_2))D_0,$$
$$\mathcal{U}_{\mathsf{hyb}}^{\mathrm{LM}}(D) = \Pi_v^\perp D_1 (D_1^\top D_1 + \varepsilon I)^{-1/2}.$$

Proposition 4.2 (Closure under composition)**.** *If $\mathcal{U}_1, \mathcal{U}_2\colon \mathbb{R}^{v\times d} \to \mathbb{R}^{v\times d}$ are both left-permutation and right-orthogonal equivariant, then $\mathcal{U}_2 \circ \mathcal{U}_1$ is also left-permutation and right-orthogonal equivariant.*

Definition 4.2 (Hybrid LPRO-equivariant maps)**.** A map $\mathcal{U}$ is called a *hybrid LPRO-equivariant map* if it is a finite composition of right-spectral maps and row-norm maps. We denote the set of such maps by $\mathcal{U}_{\mathsf{hyb}}^{v\times d}$.

Accordingly, LPRO-compatible optimizers for embeddings are obtained by applying maps in $\mathcal{U}_{\mathsf{R}}^{v\times d}$, $\mathcal{U}_{\mathsf{row}}^{v\times d}$, or $\mathcal{U}_{\mathsf{hyb}}^{v\times d}$ to update directions that transform in the same way as the gradient, such as momentum buffers formed from past gradients. For untied LM heads, the corresponding optimizer should additionally be horizontal with respect to the shared-logit-shift quotient, which is enforced by the projections $\Pi_v^\perp$ above.

Remark 4.1. Unlike the bi-orthogonal case, LPRO equivariance does not characterize a single spectral form. Right-spectral updates are a canonical global subclass, but row-norm and hybrid maps are also symmetry-compatible. This distinction is important: embeddings and LM heads have a discrete vocabulary symmetry on the left, not a full continuous orthogonal symmetry, so row-aware operations are allowed by the layer geometry. For LM heads, softmax shift invariance imposes an additional horizontal constraint; the actual update should be projected back to the quotient space whenever rowwise nonlinear operations are used.

Examples. Several recent optimizers can be interpreted through this framework. SCALE [48] applies column normalization to the EMA momentum for LM heads; under our row-vocabulary convention, this corresponds to a row-norm-based update. Other row- or column-norm-based optimizers include RMNP [33], REG [100] and NORA [161]. Finally, NORMUON [93], MUON+ [166], and MUONEQ [24], which apply row-wise and/or column-wise normalization to the orthogonal polar factor of the EMA momentum, can be viewed as hybrid spectral/row-norm optimizers.

The above equivariance principle can be generalized to other matrix parameters with different underlying symmetry structures, leading to a symmetry-to-optimizer principle and architecture–optimizer co-design detailed in Section H. In Section F and Section G, we illustrate this through deriving *intermediate-neuron permutation equivariant* optimizers for SwiGLU MLPs and *expert-permutation equivariant and shared-row-shift invariant* optimizers for MoE router weights respectively.

Practical optimizers for embeddings and LM heads. We now introduce RIGHTPOLARGRADM, which is a right-spectral optimizer similar to POLARGRAD with nuclear-norm scaling based on the right Gram matrix of the EMA momentum. With the momentum $M_k = \beta M_{k-1} + (1-\beta)G_k$, its right Gram matrix is $C_k = M_k^\top M_k$, RIGHTPOLARGRADM takes the update $W_{k+1} = (1 - \gamma_k\lambda)W_k - \gamma_k \nu_k^\alpha M_k R_k$, where $R_k = (C_k + \varepsilon I_d)^{-1/2}$ and $\nu_k = \max\{\langle\!\langle C_k, R_k \rangle\!\rangle_{\mathrm{F}}, \varepsilon\}$ for some small $\varepsilon > 0$, $\alpha \in [0, 1]$, and $\lambda > 0$ is the decoupled weight decay. ROWNORMM is the row-norm-based optimizer with $D_k = M_k$ and $\eta(t) = 1/(t + \varepsilon)$ for some small $\varepsilon > 0$. HYBRIDPOLARGRADM is the hybrid LRPO-equivariant optimizer by composing the right-spectral map and the row-norm-based map on the momentum. Their exact forms and implementation details are given in Section J.

5 Numerical Experiments on Downsized gpt-oss Pre-Training

We validate the proposed symmetry-compatible optimizer classes on four open-weight dense and MoE language models spanning different vocabulary sizes and hidden sizes, and hence substantially different embedding and LM head matrix sizes, as well as number of model parameters. We only present the results of downsized gpt-oss-20b pre-training here and leave those of other models in Section K. Our goal is to validate the practical implications of the symmetry- and geometry-based design principles developed above across architectures with distinct matrix-parameter geometries. Because our focus is on optimizer behavior rather than scaling-law-optimal pre-training, we do not train the models for the large token budgets typically prescribed by scaling laws. We pre-train all models on the 10B-token subset of FineWeb-Edu [103]. We use MUON [73] with POLAR EXPRESS coefficients [5] for all hidden and attention matrices, and ADAMW [101] for scalars and vectors. For mathematical correctness, when fused attention weight matrices are used, we apply a variant of POLAR EXPRESS that computes the orthogonal polar factor of the momentum of each fused weight matrix separately, in a similar vein to MUON SPLIT [50]. We use untied input embeddings and output LM head weights, which allows us to assign different optimizers to these two large vocabulary-indexed matrices.

We pre-train a downsized variant of gpt-oss-20b [112], which is a sparse Mixture-of-Experts (MoE) model also widely used architecture in recent open-weight language models [30, 31, 50, 55, 69, 107, 120, 136]. The model has vocabulary size 201,088, making the embedding and LM head matrices substantially large. We downsize gpt-oss-20b by reducing the number of hidden layers from 24 to 12, the hidden and intermediate dimensions from 2880 to 2048, and the number of experts from 32 to 16. This yields a total of 3,467,779,008 trainable parameters. We do not use the auxiliary load-balancing loss [129] and the router z-loss [173] in order to reduce confounding effects from auxiliary objectives and isolate the effect of optimizer geometry. Full experimental details are given in Section L. Code is available at `https://github.com/timlautk/equivariant_optimizers`.

For matrix-valued parameters other than the hidden and attention matrices, we compare four optimizer assignments: (i) ROWNORMM for embeddings, LM head, and routers, (ii) ROWNORMM for embeddings and LM head, and LEFTPOLARGRADM for routers, (iii) ROWNORMM for embeddings and LM head, and ADAMW for routers, (iv) ADAMW for embeddings, LM head, and routers. We choose ROWNORMM for the embedding and LM head matrices because it adds minimal computational overhead relative to ADAMW while preserving a symmetry-compatible geometry for vocabulary-indexed matrices. By contrast, RIGHTPOLARGRADM and HYBRIDPOLARGRADM require numerical polar decomposition, which is computationally more demanding and may require

higher numerical precision for large vocabulary matrices. We leave broader ablations over these alternatives to future work.

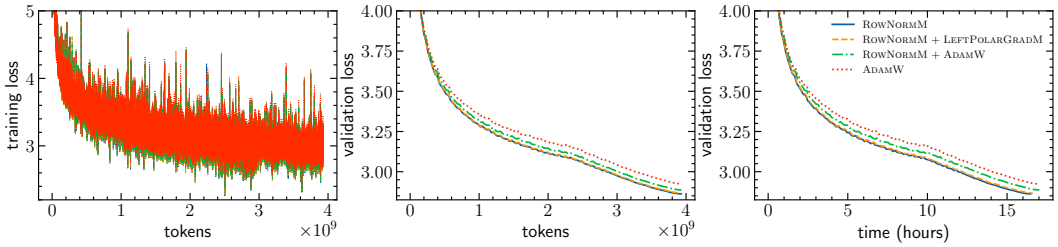


Figure 2: Training and validation losses for downsized gpt-oss pre-training. The configurations differ in the optimizers assigned to the embedding, LM head, and router matrices.

The final validation losses for configurations (i)–(iv) are 2.8617, 2.8665, 2.8862, and 2.9238, respectively. As in the OLMoE experiment, the fully coordinate-wise baseline in configuration (iv), which uses AdamW for the embedding, LM head, and router matrices, obtains the worst final validation loss. The three configurations using RowNormM for the embedding and LM head matrices all substantially improve over this baseline, suggesting that the benefit of symmetry-compatible updates for vocabulary-indexed matrices persists in a distinct sparse MoE architecture.

Among configurations (i)–(iii), the differences are smaller. As shown in Figure 2, configuration (i), which uses RowNormM for embeddings, LM head, and routers, achieves the lowest final validation loss, followed closely by configuration (ii), which uses LeftPolarGradM for routers. Configuration (iii), which keeps RowNormM for embeddings and LM head but uses AdamW for routers, is slightly worse than both geometry-compatible router variants. This ordering is consistent with the view that router geometry can matter, although the smaller gap between configurations (i)–(iii) suggests that the dominant improvement in this setting comes from replacing AdamW on the large vocabulary-indexed matrices.

Additional router diagnostics in Section K.4 show that symmetry-compatible router updates reduce load imbalance and router z-loss relative to the AdamW router baseline, both with and without auxiliary router losses.

6 Concluding Remarks

A wide range of pathological phenomena and training instabilities in large-scale neural network optimization may plausibly be attributed, at least in part, to the use of coordinate-wise adaptive gradient optimizers on matrix-valued parameters. Although Adam(W) remains the default optimizer for most neural network training tasks, largely due to legacy and practical momentum, we argue that a layerwise geometry-aware perspective is needed to improve our understanding of deep learning optimization for matrix parameters. Such a perspective is standard in the matrix optimization literature, yet remains largely absent from mainstream deep learning practice.

We establish a symmetry-induced equivariance(-invariance) principle for designing optimizers for different types of matrix parameters in neural networks. In particular, for embedding and LM head layers, which are still updated with AdamW in most training runs, this principle leads to a new class of *left-permutation-right-orthogonal equivariant* (and *shared-logit-shift invariant*) optimizers. More broadly, our results support a theoretically principled viewpoint: much of modern deep learning still relies on optimizers whose update geometry fundamentally mismatches the geometry of the layers they train. This mismatch may affect robustness, stability, scalability, and interpretability, since coordinate-wise adaptivity amplifies idiosyncrasies of parameterization by effectively optimizing in a pathological augmented space. By contrast, geometry-aware updates offer the prospect of more stable, parameterization-invariant, and theoretically grounded training procedures. In short, introducing symmetry and equivariance into optimizer design suggests a new direction for advancing the science of large-scale model pre-training.

Acknowledgments and Disclosure of Funding

This work was supported by computational resources from Prime Intellect.

References

[1] E. Abbe and E. Boix-Adsera. On the non-universality of deep learning: quantifying the cost of symmetry. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[2] K. Ahn, N. Amsel, and J. Langford. Dion2: A simple method to shrink matrix in Muon. *arXiv preprint arXiv:2512.16928*, 2025.

[3] K. Ahn, B. Xu, N. Abreu, Y. Fan, G. Magakyan, P. Sharma, Z. Zhan, and J. Langford. Dion: Distributed orthonormalized updates. *arXiv preprint arXiv:2504.05295*, 2025.

[4] J. Ainslie, J. Lee-Thorp, M. De Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[5] N. Amsel, D. Persson, C. Musco, and R. M. Gower. The Polar Express: Optimal matrix sign methods and their application to the Muon algorithm. In *International Conference on Learning Representations (ICLR)*, 2026.

[6] K. An, Y. Liu, R. Pan, S. Ma, D. Goldfarb, and T. Zhang. ASGO: Adaptive structured gradient optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[7] R. Anil, V. Gupta, T. Koren, K. Regan, and Y. Singer. Scalable second order optimization for deep learning. *arXiv preprint arXiv:2002.09018*, 2020.

[8] Q. Anthony, Y. Tokpanov, S. Szot, S. Rajagopal, P. Medepalli, R. Iyer, V. Shyam, A. Golubeva, A. Chaurasia, X. Yang, T. Figliolia, R. Washbourne, D. Thorstensen, A. Pearson, Z. Grossbart, J. van Patten, E. Barsoum, Z. Gu, Y. Fu, and B. Millidge. Training foundation models on a full-stack AMD platform: Compute, networking, and system design. *arXiv preprint arXiv:2511.17127*, 2025.

[9] L. Autonne. Sur les groupes linéaires, réels et orthogonaux. *Bulletin de la Société Mathématique de France*, 30:121–134, 1902.

[10] J. Bernstein. Deriving Muon. https://jeremybernste.in/writing/deriving-muon, 2025.

[11] J. Bernstein. Modular manifolds. Thinking Machines Lab: Connectionism, 2025. URL https://thinkingmachines.ai/blog/modular-manifolds/.

[12] J. Bernstein and L. Newhouse. Old optimizer, new norm: An anthology. In *Annual Workshop on Optimization for Machine Learning (OPT)*, 2024.

[13] J. Bernstein and L. Newhouse. Modular duality in deep learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025.

[14] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar. signSGD: Compressed optimisation for non-convex problems. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[15] R. Bhatia. *Matrix Analysis*, volume 169. Springer Science & Business Media, 2013.

[16] T. Boissin, T. Massena, F. Mamalet, and M. Serrurier. Turbo-Muon: Accelerating orthogonality-based optimization with pre-conditioning. *arXiv preprint arXiv:2512.04632*, 2025.

[17] F. Bordes, R. Y. Pang, A. Ajay, A. C. Li, A. Bardes, S. Petryk, O. Mañas, Z. Lin, A. Mahmoud, B. Jayaraman, M. Ibrahim, M. Hall, Y. Xiong, J. Lebensold, C. Ross, S. Jayakumar, C. Guo, D. Bouchacourt, H. Al-Tahan, K. Padthe, V. Sharma, H. Xu, X. E. Tan, M. Richards, S. Lavoie, P. Astolfi, R. A. Hemmat, J. Chen, K. Tirumala, R. Assouel, M. Moayeri, A. Talattof, K. Chaudhuri, Z. Liu, X. Chen, Q. Garrido, K. Ullrich, A. Agrawal, K. Saenko, A. Celikyilmaz, and V. Chandra. An introduction to vision-language modeling. *arXiv preprint arXiv:2405.17247*, 2024.

[18] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[19] S. Buchanan. A faster manifold Muon with ADMM. https://sdbuchanan.com/blog/manifold-muon/, 2025.

[20] D. Carlson, V. Cevher, and L. Carin. Stochastic spectral descent for restricted Boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015.

[21] D. Carlson, E. Collins, Y.-P. Hsieh, L. Carin, and V. Cevher. Preconditioned spectral descent for deep learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[22] D. Carlson, Y.-P. Hsieh, E. Collins, L. Carin, and V. Cevher. Stochastic spectral descent for discrete graphical models. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):296–311, 2016.

[23] D. Chang, Y. Liu, and G. Yuan. On the convergence of Muon and beyond. *arXiv preprint arXiv:2509.15816*, 2025.

[24] D. Chang, Q. Shi, L. Zhang, Y. Li, R. Zhang, Y. Lu, Y. Liu, and G. Yuan. MuonEq: Balancing before orthogonalization with lightweight equilibration. *arXiv preprint arXiv:2603.28254*, 2026.

[25] L. Chen, J. Li, and Q. Liu. Muon optimizes under spectral norm constraints. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856.

[26] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel. PaLM: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[27] M. Crawshaw, C. Modi, M. Liu, and R. M. Gower. An exploration of non-Euclidean gradient descent: Muon and its many variants. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

[28] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

[29] D. Davis and D. Drusvyatskiy. When do spectral gradient updates help in deep learning? *arXiv preprint arXiv:2512.04299*, 2025.

[30] DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence, 2026.

[31] DeepSeek-AI, A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Guo, D. Yang, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Bao, H. Xu, H. Wang, H. Zhang, H. Ding, H. Xin, H. Gao, H. Li, H. Qu, J. L. Cai, J. Liang, J. Guo, J. Ni, J. Li, J. Wang, J. Chen, J. Chen, J. Yuan, J. Qiu, J. Li, J. Song, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Xu, L. Xia, L. Zhao, L. Wang, L. Zhang, M. Li, M. Wang, M. Zhang, M. Zhang, M. Tang, M. Li, N. Tian, P. Huang, P. Wang, P. Zhang, Q. Wang, Q. Zhu, Q. Chen, Q. Du, R. J. Chen, R. L. Jin, R. Ge, R. Zhang, R. Pan, R. Wang, R. Xu, R. Zhang, R. Chen, S. S. Li, S. Lu, S. Zhou, S. Chen, S. Wu, S. Ye, S. Ye, S. Ma, S. Wang, S. Zhou, S. Yu, S. Zhou, S. Pan, T. Wang, T. Yun, T. Pei, T. Sun, W. L. Xiao, W. Zeng, W. Zhao, W. An, W. Liu, W. Liang, W. Gao, W. Yu, W. Zhang, X. Q. Li, X. Jin, X. Wang, X. Bi, X. Liu, X. Wang, X. Shen, X. Chen, X. Zhang, X. Chen, X. Nie, X. Sun, X. Wang, X. Cheng, X. Liu, X. Xie, X. Liu, X. Yu, X. Song, X. Shan, X. Zhou, X. Yang, X. Li, X. Su, X. Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Y. Zhang, Y. Xu, Y. Xu, Y. Huang, Y. Li, Y. Zhao, Y. Sun, Y. Li, Y. Wang, Y. Yu, Y. Zheng, Y. Zhang, Y. Shi, Y. Xiong, Y. He, Y. Tang, Y. Piao, Y. Wang, Y. Tan, Y. Ma, Y. Liu, Y. Guo, Y. Wu, Y. Ou, Y. Zhu, Y. Wang, Y. Gong, Y. Zou, Y. He, Y. Zha, Y. Xiong, Y. Ma, Y. Yan, Y. Luo, Y. You, Y. Liu, Y. Zhou, Z. F. Wu, Z. Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Xu, Z. Huang, Z. Zhang, Z. Xie, Z. Zhang, Z. Hao, Z. Gou, Z. Ma, Z. Yan, Z. Shao, Z. Xu, Z. Wu, Z. Zhang, Z. Li, Z. Gu, Z. Zhu, Z. Liu, Z. Li, Z. Xie, Z. Song, Z. Gao, and Z. Pan. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[32] M. Dehghani, J. Djolonga, B. Mustafa, P. Padlewski, J. Heek, J. Gilmer, A. Steiner, M. Caron, R. Geirhos, I. Alabdulmohsin, R. Jenatton, L. Beyer, M. Tschannen, A. Arnab, X. Wang, C. Riquelme, M. Minderer, J. Puigcerver, U. Evci, M. Kumar, S. van Steenkiste, G. F. Elsayed, A. Mahendran, F. Yu, A. Oliver, F. Huot, J. Bastings, M. P. Collier, A. Gritsenko, V. Birodkar, C. Vasconcelos, Y. Tay, T. Mensink, A. Kolesnikov, F. Pavetić, D. Tran, T. Kipf, M. Lučić, X. Zhai, D. Keysers, J. Harmsen, and N. Houlsby. Scaling vision transformers to 22 billion parameters. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2023.

[33] S. Deng, Z. Ouyang, T. Pang, Z. Liu, R. Jin, S. Yu, and Y. Yang. RMNP: Row-momentum normalized preconditioning for scalable matrix-based optimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

[34] A. Dewulf, D. Pai, L. Yang, A. Zhang, and B. Keigwin. Aurora: A leverage-aware optimizer for rectangular matrices, May 2026. URL https://blog.tilderesearch.com/blog/aurora.

[35] C. Ding, D. Sun, J. Sun, and K.-C. Toh. Spectral operators of matrices. *Mathematical Programming*, 168 (1):509–531, 2018.

[36] C. Ding, D. Sun, J. Sun, and K.-C. Toh. Spectral operators of matrices: Semismoothness and characterizations of the generalized Jacobian. *SIAM Journal on Optimization*, 30(1):630–659, 2020.

[37] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

[38] Z. Du and W. Su. The Newton–Muon optimizer. *arXiv preprint arXiv:2604.01472*, 2026.

[39] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[40] R. Eschenhagen, A. Immer, R. Turner, F. Schneider, and P. Hennig. Kronecker-factored approximate curvature for modern neural network architectures. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[41] R. Eschenhagen, A. Cai, T.-H. Lee, and H.-J. M. Shi. Clarifying Shampoo: Adapting spectral descent to stochasticity and the parameter trajectory. *arXiv preprint arXiv:2602.09314*, 2026.

[42] Essential AI. Layer sharding for large-scale training with Muon. https://www.essential.ai/research/infra, 2025.

[43] Essential AI, I. Shah, A. M. Polloreno, K. Stratos, P. Monk, A. Chaluvaraju, A. Hojel, A. Ma, A. Thomas, A. Tanwer, D. J. Shah, K. Nguyen, K. Smith, M. Callahan, M. Pust, M. Parmar, P. Rushton, P. Mazarakis, R. Kapila, S. Srivastava, S. Singla, T. Romanski, Y. Vanjani, and A. Vaswani. Practical efficiency of Muon for pretraining. *arXiv preprint arXiv:2505.02222*, 2025.

[44] O. Filatov, J. Wang, J. Ebert, and S. Kesselheim. Optimal scaling needs optimal norm. *arXiv preprint arXiv:2510.03871*, 2025.

[45] K. Frans, S. Levine, and P. Abbeel. A stable whitening optimizer for efficient neural network training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[46] Gemma Team, M. Riviere, S. Pathak, P. G. Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé, J. Ferret, P. Liu, P. Tafti, A. Friesen, M. Casbon, S. Ramos, R. Kumar, C. L. Lan, S. Jerome, A. Tsitsulin, N. Vieillard, P. Stanczyk, S. Girgin, N. Momchev, M. Hoffman, S. Thakoor, J.-B. Grill, B. Neyshabur, O. Bachem, A. Walton, A. Severyn, A. Parrish, A. Ahmad, A. Hutchison, A. Abdagic, A. Carl, A. Shen, A. Brock, A. Coenen, A. Laforge, A. Paterson, B. Bastian, B. Piot, B. Wu, B. Royal, C. Chen, C. Kumar, C. Perry, C. Welty, C. A. Choquette-Choo, D. Sinopalnikov, D. Weinberger, D. Vijaykumar, D. Rogozińska, D. Herbison, E. Bandy, E. Wang, E. Noland, E. Moreira, E. Senter, E. Eltyshev, F. Visin, G. Rasskin, G. Wei, G. Cameron, G. Martins, H. Hashemi, H. Klimczak-Plucińska, H. Batra, H. Dhand, I. Nardini, J. Mein, J. Zhou, J. Svensson, J. Stanway, J. Chan, J. P. Zhou, J. Carrasqueira, J. Iljazi, J. Becker, J. Fernandez, J. van Amersfoort, J. Gordon, J. Lipschultz, J. Newlan, J. yeong Ji, K. Mohamed, K. Badola, K. Black, K. Millican, K. McDonell, K. Nguyen, K. Sodhia, K. Greene, L. L. Sjoesund, L. Usui, L. Sifre, L. Heuermann, L. Lago, L. McNealus, L. B. Soares, L. Kilpatrick, L. Dixon, L. Martins, M. Reid, M. Singh, M. Iverson, M. Görner, M. Velloso, M. Wirth, M. Davidow, M. Miller, M. Rahtz, M. Watson, M. Risdal, M. Kazemi, M. Moynihan, M. Zhang, M. Kahng, M. Park, M. Rahman, M. Khatwani, N. Dao, N. Bardoliwalla, N. Devanathan, N. Dumai, N. Chauhan, O. Wahltinez, P. Botarda, P. Barnes, P. Barham, P. Michel, P. Jin, P. Georgiev, P. Culliton, P. Kuppala, R. Comanescu, R. Merhej, R. Jana, R. A. Rokni, R. Agarwal, R. Mullins, S. Saadat, S. M. Carthy, S. Cogan, S. Perrin, S. M. R. Arnold, S. Krause, S. Dai, S. Garg, S. Sheth, S. Ronstrom, S. Chan, T. Jordan, T. Yu, T. Eccles, T. Hennigan, T. Kocisky, T. Doshi, V. Jain, V. Yadav, V. Meshram, V. Dharmadhikari, W. Barkley, W. Wei, W. Ye, W. Han, W. Kwon, X. Xu, Z. Shen, Z. Gong, Z. Wei, V. Cotruta, P. Kirk, A. Rao, M. Giang, L. Peran, T. Warkentin, E. Collins, J. Barral, Z. Ghahramani, R. Hadsell, D. Sculley, J. Banks, A. Dragan, S. Petrov, O. Vinyals, J. Dean, D. Hassabis, K. Kavukcuoglu, C. Farabet, E. Buchatskaya, S. Borgeaud, N. Fiedel, A. Joulin, K. Kenealy, R. Dadashi, and A. Andreev. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

[47] Gemma Team, A. Kamath, J. Ferret, S. Pathak, N. Vieillard, R. Merhej, S. Perrin, T. Matejovicova, A. Ramé, M. Rivière, L. Rouillard, T. Mesnard, G. Cideron, J.-B. Grill, S. Ramos, E. Yvinec, M. Casbon, E. Pot, I. Penchev, G. Liu, F. Visin, K. Kenealy, L. Beyer, X. Zhai, A. Tsitsulin, R. Busa-Fekete, A. Feng, N. Sachdeva, B. Coleman, Y. Gao, B. Mustafa, I. Barr, E. Parisotto, D. Tian, M. Eyal, C. Cherry, J.-T. Peter, D. Sinopalnikov, S. Bhupatiraju, R. Agarwal, M. Kazemi, D. Malkin, R. Kumar, D. Vilar, I. Brusilovsky, J. Luo, A. Steiner, A. Friesen, A. Sharma, A. Sharma, A. M. Gilady, A. Goedeckemeyer, A. Saade, A. Feng, A. Kolesnikov, A. Bendebury, A. Abdagic, A. Vadi, A. György, A. S. Pinto, A. Das, A. Bapna, A. Miech, A. Yang, A. Paterson, A. Shenoy, A. Chakrabarti, B. Piot, B. Wu, B. Shahriari, B. Petrini, C. Chen, C. L. Lan, C. A. Choquette-Choo, C. Carey, C. Brick, D. Deutsch, D. Eisenbud, D. Cattle, D. Cheng, D. Paparas, D. S. Sreepathihalli, D. Reid, D. Tran, D. Zelle, E. Noland, E. Huizenga, E. Kharitonov, F. Liu, G. Amirkhanyan, G. Cameron, H. Hashemi, H. Klimczak-Plucińska, H. Singh, H. Mehta, H. T. Lehri, H. Hazimeh, I. Ballantyne, I. Szpektor, I. Nardini, J. Pouget-Abadie, J. Chan, J. Stanton, J. Wieting, J. Lai, J. Orbay, J. Fernandez, J. Newlan, J. yeong Ji, J. Singh, K. Black, K. Yu, K. Hui, K. Vodrahalli, K. Greff, L. Qiu, M. Valentine, M. Coelho, M. Ritter, M. Hoffman, M. Watson, M. Chaturvedi, M. Moynihan, M. Ma, N. Babar, N. Noy, N. Byrd, N. Roy, N. Momchev, N. Chauhan, N. Sachdeva, O. Bunyan, P. Botarda, P. Caron, P. K. Rubenstein, P. Culliton, P. Schmid, P. G. Sessa, P. Xu, P. Stanczyk, P. Tafti, R. Shivanna, R. Wu, R. Pan, R. Rokni, R. Willoughby, R. Vallu, R. Mullins, S. Jerome, S. Smoot, S. Girgin, S. Iqbal, S. Reddy, S. Sheth, S. Põder, S. Bhatnagar, S. R. Panyam, S. Eiger, S. Zhang, T. Liu, T. Yacovone, T. Liechty, U. Kalra, U. Evci, V. Misra, V. Roseberry, V. Feinberg, V. Kolesnikov, W. Han, W. Kwon, X. Chen, Y. Chow, Y. Zhu, Z. Wei, Z. Egyed, V. Cotruta, M. Giang, P. Kirk, A. Rao, K. Black, N. Babar, J. Lo, E. Moreira, L. G. Martins, O. Sanseviero, L. Gonzalez, Z. Gleicher, T. Warkentin, V. Mirrokni, E. Senter, E. Collins, J. Barral, Z. Ghahramani, R. Hadsell, Y. Matias, D. Sculley, S. Petrov, N. Fiedel, N. Shazeer, O. Vinyals, J. Dean, D. Hassabis, K. Kavukcuoglu, C. Farabet, E. Buchatskaya, J.-B. Alayrac, R. Anil, Dmitry, Lepikhin, S. Borgeaud, O. Bachem, A. Joulin, A. Andreev, C. Hardin, R. Dadashi, and L. Hussenot. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

[48] A. Glentis, J. Li, A. Han, and M. Hong. Memory-efficient LLM pretraining via minimalist optimizer design. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

[49] GLM-4.5 Team, A. Zeng, X. Lv, Q. Zheng, Z. Hou, B. Chen, C. Xie, C. Wang, D. Yin, H. Zeng, J. Zhang, K. Wang, L. Zhong, M. Liu, R. Lu, S. Cao, X. Zhang, X. Huang, Y. Wei, Y. Cheng, Y. An, Y. Niu, Y. Wen, Y. Bai, Z. Du, Z. Wang, Z. Zhu, B. Zhang, B. Wen, B. Wu, B. Xu, C. Huang, C. Zhao, C. Cai, C. Yu, C. Li, C. Ge, C. Huang, C. Zhang, C. Xu, C. Zhu, C. Li, C. Yin, D. Lin, D. Yang, D. Jiang, D. Ai, E. Zhu, F. Wang, G. Pan, G. Wang, H. Sun, H. Li, H. Li, H. Hu, H. Zhang, H. Peng, H. Tai, H. Zhang, H. Wang, H. Yang, H. Liu, H. Zhao, H. Liu, H. Yan, H. Liu, H. Chen, J. Li, J. Zhao, J. Ren, J. Jiao, J. Zhao, J. Yan, J. Wang, J. Gui, J. Zhao, J. Liu, J. Li, J. Li, J. Lu, J. Wang, J. Yuan, J. Li, J. Du, J. Du, J. Liu, J. Zhi, J. Gao, K. Wang, L. Yang, L. Xu, L. Fan, L. Wu, L. Ding, L. Wang, M. Zhang, M. Li, M. Xu, M. Zhao, M. Zhai, P. Du, Q. Dong, S. Lei, S. Tu, S. Yang, S. Lu, S. Li, S. Li, Shuang-Li, S. Yang, S. Yi, T. Yu, W. Tian, W. Wang, W. Yu, W. L. Tam, W. Liang, W. Liu, X. Wang, X. Jia, X. Gu, X. Ling, X. Wang, X. Fan, X. Pan, X. Zhang, X. Zhang, X. Fu, X. Zhang, Y. Xu, Y. Wu, Y. Lu, Y. Wang, Y. Zhou, Y. Pan, Y. Zhang, Y. Wang, Y. Li, Y. Su, Y. Geng, Y. Zhu, Y. Yang, Y. Li, Y. Wu, Y. Li, Y. Liu, Y. Wang, Y. Li, Y. Zhang, Z. Liu, Z. Yang, Z. Zhou, Z. Qiao, Z. Feng, Z. Liu, Z. Zhang, Z. Wang, Z. Yao, Z. Wang, Z. Liu, Z. Chai, Z. Li, Z. Zhao, W. Chen, J. Zhai, B. Xu, M. Huang, H. Wang, J. Li, Y. Dong, and J. Tang. GLM-4.5: Agentic, reasoning, and coding (ARC) foundation models. *arXiv preprint arXiv:2508.06471*, 2025.

[50] GLM-5 Team, A. Zeng, X. Lv, Z. Hou, Z. Du, Q. Zheng, B. Chen, D. Yin, C. Ge, C. Huang, C. Xie, C. Zhu, C. Yin, C. Wang, G. Pan, H. Zeng, H. Zhang, H. Wang, H. Chen, J. Zhang, J. Jiao, J. Guo, J. Wang, J. Du, J. Wu, K. Wang, L. Li, L. Fan, L. Zhong, M. Liu, M. Zhao, P. Du, Q. Dong, R. Lu, Shuang-Li, S. Cao, S. Liu, T. Jiang, X. Chen, X. Zhang, X. Huang, X. Dong, Y. Xu, Y. Wei, Y. An, Y. Niu, Y. Zhu, Y. Wen, Y. Cen, Y. Bai, Z. Qiao, Z. Wang, Z. Wang, Z. Zhu, Z. Liu, Z. Li, B. Wang, B. Wen, C. Huang, C. Cai, C. Yu, C. Li, C. Hu, C. Zhang, D. Zhang, D. Lin, D. Yang, D. Wang, D. Ai, E. Zhu, F. Yi, F. Chen, G. Wen, H. Sun, H. Zhao, H. Hu, H. Zhang, H. Liu, H. Zhang, H. Peng, H. Tai, H. Zhang, H. Liu, H. Wang, H. Yan, H. Ge, H. Liu, H. Chu, J. Zhao, J. Wang, J. Zhao, J. Ren, J. Wang, J. Zhang, J. Gui, J. Zhao, J. Li, J. An, J. Li, J. Yuan, J. Du, J. Liu, J. Zhi, J. Duan, K. Zhou, K. Wei, K. Wang, K. Luo, L. Zhang, L. Sha, L. Xu, L. Wu, L. Ding, L. Chen, M. Li, N. Lin, P. Ta, Q. Zou, R. Song, R. Yang, S. Tu, S. Yang, S. Wu, S. Zhang, S. Li, S. Li, S. Fan, W. Qin, W. Tian, W. Zhang, W. Yu, W. Liang, X. Kuang, X. Cheng, X. Li, X. Yan, X. Hu, X. Ling, X. Fan, X. Xia, X. Zhang, X. Zhang, X. Pan, X. Zou, X. Zhang, Y. Liu, Y. Wu, Y. Li, Y. Wang, Y. Zhu, Y. Tan, Y. Zhou, Y. Pan, Y. Zhang, Y. Su, Y. Geng, Y. Yan, Y. Tan, Y. Bi, Y. Shen, Y. Yang, Y. Li, Y. Liu, Y. Wang, Y. Li, Y. Wu, Y. Zhang, Y. Duan, Y. Zhang, Z. Liu, Z. Jiang, Z. Yan, Z. Zhang, Z. Wei, Z. Chen, Z. Feng, Z. Yao, Z. Chai, Z. Wang, Z. Zhang, B. Xu, M. Huang, H. Wang, J. Li, Y. Dong, and J. Tang. GLM-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

[51] D. Goldfarb, Y. Ren, and A. Bahamou. Practical quasi-Newton methods for training deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[52] S. Gong, S. Agarwal, Y. Zhang, J. Ye, L. Zheng, M. Li, C. An, P. Zhao, W. Bi, J. Han, H. Peng, and L. Kong. Scaling diffusion language models via adaptation from autoregressive models. In *International Conference on Learning Representations (ICLR)*, 2025.

[53] W. Gong, J. Zazo, Q. Luo, P. Wang, J. Hensman, and C. Ma. ARO: A new lens on matrix optimization for large models. *arXiv preprint arXiv:2602.09006*, 2026.

[54] A. Gonon, A.-A. Muşat, and N. Boumal. Insights on Muon from simple quadratics. *arXiv preprint arXiv:2602.11948*, 2026.

[55] Google DeepMind. Gemma 4 model card, April 2026. URL https://ai.google.dev/gemma/docs/core/model_card_4.

[56] E. Grishina, M. Smirnov, and M. Rakhuba. Accelerating Newton-Schulz iteration for orthogonalization via Chebyshev-type polynomials. *arXiv preprint arXiv:2506.10935*, 2025.

[57] Y. Gu and Z. Xie. Mano: Restriking manifold optimization for LLM training. *arXiv preprint arXiv:2601.23000*, 2026.

[58] A. Gupta, R. Celente, A. Shivanna, D. T. Braithwaite, G. Dexter, S. Tang, H. Udagawa, D. Silva, R. Ramanath, and S. S. Keerthi. Effective quantization of Muon optimizer states. *arXiv preprint arXiv:2509.23106*, 2025.

[59] V. Gupta, T. Koren, and Y. Singer. Shampoo: Preconditioned stochastic tensor optimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[60] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

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

[62] N. J. Higham. Stable iterations for the matrix square root. *Numerical Algorithms*, 15(2):227–242, 1997.

[63] N. J. Higham. *Functions of Matrices: Theory and Computation*. Society for Industrial and Applied Mathematics, 2008.

[64] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, O. Vinyals, J. W. Rae, and L. Sifre. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[65] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1994.

[66] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 2nd edition, 2012.

[67] Y. Hu, H. Song, J. Deng, J. Wang, J. Chen, K. Zhou, Y. Zhu, J. Jiang, Z. Dong, W. X. Zhao, and J.-R. Wen. YuLan-Mini: Pushing the limits of open data-efficient language model. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), Long Papers*, 2025.

[68] F. Huang, Y. Luo, and S. Chen. LiMuon: Light and fast Muon optimizer for large models. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

[69] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. de las Casas, E. B. Hanna, F. Bressand, G. Lengyel, G. Bour, G. Lample, L. R. Lavaud, L. Saulnier, M.-A. Lachaux, P. Stock, S. Subramanian, S. Yang, S. Antoniak, T. L. Scao, T. Gervet, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[70] R. Jiang, Z. Mhammedi, M. Mohri, and A. Mokhtari. Adaptive matrix online learning through smoothing with guarantees for nonsmooth nonconvex optimization. In *Proceedings of the Conference on Learning Theory (COLT)*, 2026.

[71] Z. Jiang, J. Gu, H. Zhu, and D. Pan. Pre-RMSNorm and Pre-CRMSNorm transformers: equivalent and efficient Pre-LN transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[72] K. Jordan, J. Bernstein, B. Rappazzo, @fernbear.bsky.social, B. Vlado, Y. Jiacheng, F. Cesista, B. Koszarsky, and @Grad62304977. `modded-nanogpt`: Speedrunning the NanoGPT baseline, 2024. URL https://github.com/KellerJordan/modded-nanogpt.

[73] K. Jordan, Y. Jin, V. Boza, J. You, F. Cesista, L. Newhouse, and J. Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

[74] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[75] G. Y. Kim and M.-h. Oh. Convergence of Muon with Newton-Schulz. In *International Conference on Learning Representations (ICLR)*, 2026.

[76] Kimi Team. Kimi K2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

[77] D. P. Kingma and J. L. Ba. Adam: a method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[78] D. Kovalev. Understanding gradient orthogonalization for deep learning via non-Euclidean trust-region optimization. *arXiv preprint arXiv:2503.12645*, 2025.

[79] D. Kovalev and E. Borodich. Non-Euclidean SGD for structured optimization: Unified analysis and improved rates. *arXiv preprint arXiv:2511.11466*, 2025.

[80] A. Kravatskiy, I. Kozyrev, N. Kozlov, A. Vinogradov, D. Merkulov, and I. Oseledets. The Ky Fan norms and beyond: Dual norms and combinations for matrix optimization. *arXiv preprint arXiv:2512.09678*, 2025.

[81] T. Large, Y. Liu, M. Huh, H. Bahng, P. Isola, and J. Bernstein. Scalable optimization in the modular norm. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[82] T. T.-K. Lau, J. Zeng, B. Wu, and Y. Yao. A proximal block coordinate descent algorithm for deep neural network training. In *International Conference on Learning Representations (ICLR), Workshop Track*, 2018.

[83] T. T.-K. Lau, Q. Long, and W. Su. POLARGRAD: A class of matrix-gradient optimizers from a unifying preconditioning perspective. *arXiv preprint arXiv:2505.21799*, 2025.

[84] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, and Z. Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations (ICLR)*, 2021.

[85] A. S. Lewis. The convex analysis of unitarily invariant matrix functions. *Journal of Convex Analysis*, 2 (1):173–183, 1995.

[86] A. S. Lewis. Group invariance and convex matrix analysis. *SIAM Journal on Matrix Analysis and Applications*, 17(4):927–949, 1996.

[87] A. S. Lewis. The mathematics of eigenvalue optimization. *Mathematical Programming*, 97:155–176, 2003.

[88] A. S. Lewis and M. L. Overton. Eigenvalue optimization. *Acta Numerica*, 5:149–190, 1996.

[89] H. Li, Y. Dong, and Z. Lin. Convergence rate analysis of the AdamW-style Shampoo: Unifying one-sided and two-sided preconditioning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

[90] J. Li and M. Hong. A note on the convergence of Muon. *arXiv preprint arXiv:2502.02900*, 2025.

[91] X.-L. Li. Preconditioned stochastic gradient descent. *IEEE Transactions on Neural Networks and Learning Systems*, 29(5):1454–1466, 2017.

[92] Z. Li, Y. Zhang, and S. Arora. Why are convolutional nets more sample-efficient than fully-connected nets? In *International Conference on Learning Representations (ICLR)*, 2021.

[93] Z. Li, L. Liu, C. Liang, W. Chen, and T. Zhao. NorMuon: Making Muon more efficient and scalable. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

[94] W. Liang, T. Liu, L. Wright, W. Constable, A. Gu, C.-C. Huang, I. Zhang, W. Feng, H. Huang, J. Wang, S. Purandare, G. Nadathur, and S. Idreos. TORCHTITAN: One-stop PyTorch native solution for production ready LLM pre-training. In *International Conference on Learning Representations (ICLR)*, 2025.

[95] J. Lim, S. Lee, D. Kim, T. Kim, E. Park, J. Lee, J. Lee, J. Lee, W. T. Cheung, D. Choi, J. Her, J. Huh, H. Jung, C. Kang, B. Kim, M. Kim, T. Kim, Y. Kim, H. Kweon, H. Lee, K. Lee, D. Oh, Y. Park, B. Ryu, and D. Weon. Motif 2 12.7B technical report. *arXiv preprint arXiv:2511.07464*, 2025.

[96] T. Lin. Flash-Muon: An efficient implementation of Muon optimizer, 2025. URL https://github.com/nil0x9/flash-muon.

[97] W. Lin, S. C. Lowe, F. Dangel, R. Eschenhagen, Z. Xu, and R. B. Grosse. Understanding and improving Shampoo and SOAP via Kullback–Leibler minimization. In *International Conference on Learning Representations (ICLR)*, 2026.

[98] J. Liu, J. Su, X. Yao, Z. Jiang, G. Lai, Y. Du, Y. Qin, W. Xu, E. Lu, J. Yan, Y. Chen, H. Zheng, Y. Liu, S. Liu, B. Yin, W. He, H. Zhu, Y. Wang, J. Wang, M. Dong, Z. Zhang, Y. Kang, H. Zhang, X. Xu, Y. Zhang, Y. Wu, X. Zhou, and Z. Yang. Muon is scalable for LLM training. *arXiv preprint arXiv:2502.16982*, 2025.

[99] Y. Liu, A. Yuan, and Q. Gu. MARS-M: When variance reduction meets matrices. *arXiv preprint arXiv:2510.21800*, 2025.

[100] Z. Liu, H. Wu, X. Fu, S. Liu, X. Han, T. Zhong, and M. Yuan. REG: A regularization optimizer for robust training dynamics. *arXiv preprint arXiv:2510.03691*, 2025.

[101] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[102] A. Lou, C. Meng, and S. Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

[103] A. Lozhkov, L. Ben Allal, L. von Werra, and T. Wolf. FineWeb-Edu: the finest collection of educational content, 2024. URL https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu.

[104] J. Ma, Y. Huang, Y. Chi, and Y. Chen. Preconditioning benefits of spectral orthogonalization in Muon. *arXiv preprint arXiv:2601.13474*, 2026.

[105] J. Martens and R. Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

[106] H. B. McMahan and M. Streeter. Adaptive bound optimization for online convex optimization. In *Proceedings of the Conference on Learning Theory (COLT)*, 2010.

[107] N. Muennighoff, L. Soldaini, D. Groeneveld, K. Lo, J. Morrison, S. Min, W. Shi, P. Walsh, O. Tafjord, N. Lambert, Y. Gu, S. Arora, A. Bhagia, D. Schwenk, D. Wadden, A. Wettig, B. Hui, T. Dettmers, D. Kiela, A. Farhadi, N. A. Smith, P. W. Koh, A. Singh, and H. Hajishirzi. OLMoE: Open mixture-of-experts language models. In *International Conference on Learning Representations (ICLR)*, 2025.

[108] Y. Nakatsukasa and R. W. Freund. Computing fundamental matrix decompositions accurately via the matrix sign function in two iterations: The power of Zolotarev's functions. *SIAM Review*, 58(3):461–493, 2016.

[109] Y. Nakatsukasa, Z. Bai, and F. Gygi. Optimizing Halley's iteration for computing the matrix polar decomposition. *SIAM Journal on Matrix Analysis and Applications*, 31(5):2700–2720, 2010.

[110] A. Y. Ng. Feature selection, l_1 vs. l_2 regularization, and rotational invariance. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2004.

[111] S. Nie, F. Zhu, Z. You, X. Zhang, J. Ou, J. Hu, J. Zhou, Y. Lin, J.-R. Wen, and C. Li. Large language diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[112] OpenAI, S. Agarwal, L. Ahmad, J. Ai, S. Altman, A. Applebaum, E. Arbus, R. K. Arora, Y. Bai, B. Baker, H. Bao, B. Barak, A. Bennett, T. Bertao, N. Brett, E. Brevdo, G. Brockman, S. Bubeck, C. Chang, K. Chen, M. Chen, E. Cheung, A. Clark, D. Cook, M. Dukhan, C. Dvorak, K. Fives, V. Fomenko, T. Garipov, K. Georgiev, M. Glaese, T. Gogineni, A. Goucher, L. Gross, K. G. Guzman, J. Hallman, J. Hehir, J. Heidecke, A. Helyar, H. Hu, R. Huet, J. Huh, S. Jain, Z. Johnson, C. Koch, I. Kofman, D. Kundel, J. Kwon, V. Kyrylov, E. Y. Le, G. Leclerc, J. P. Lennon, S. Lessans, M. Lezcano-Casado, Y. Li, Z. Li, J. Lin, J. Liss, Lily, Liu, J. Liu, K. Lu, C. Lu, Z. Martinovic, L. McCallum, J. McGrath, S. McKinney, A. McLaughlin, S. Mei, S. Mostovoy, T. Mu, G. Myles, A. Neitz, A. Nichol, J. Pachocki, A. Paino, D. Palmie, A. Pantuliano, G. Parascandolo, J. Park, L. Pathak, C. Paz, L. Peran, D. Pimenov, M. Pokrass, E. Proehl, H. Qiu, G. Raila, F. Raso, H. Ren, K. Richardson, D. Robinson, B. Rotsted, H. Salman, S. Sanjeev, M. Schwarzer, D. Sculley, H. Sikchi, K. Simon, K. Singhal, Y. Song, D. Stuckey,

Z. Sun, P. Tillet, S. Toizer, F. Tsimpourlas, N. Vyas, E. Wallace, X. Wang, M. Wang, O. Watkins, K. Weil, A. Wendling, K. Whinnery, C. Whitney, H. Wong, L. Yang, Y. Yang, M. Yasunaga, K. Ying, W. Zaremba, W. Zhan, C. Zhang, B. Zhang, E. Zhang, and S. Zhao. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

[113] T. Pethick, W. Xie, K. Antonakopoulos, Z. Zhu, A. Silveti-Falls, and V. Cevher. Training deep learning models with norm-constrained LMOs. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025.

[114] T. Pethick, K. Antonakopoulos, A. Silveti-Falls, L. C. Vankadara, and V. Cevher. Training neural networks at any scale: An exposition. *IEEE Signal Processing Magazine*, 43(3):21–36, 2026.

[115] O. Pooladzandi and X.-L. Li. Curvature-informed SGD via general purpose Lie-group preconditioners. *arXiv preprint arXiv:2402.04553*, 2024.

[116] Prime Intellect Team, M. Senghaas, F. Obeid, S. Jaghouar, W. Brown, J. M. Ong, D. Auras, M. Sirovatka, J. Straube, A. Baker, S. Müller, J. Mattern, M. Basra, A. Ismail, D. Scherm, C. Miller, A. Patel, S. Kirsten, M. Sieg, C. Reetz, K. Erdem, V. Weisser, and J. Hagemann. INTELLECT-3: Technical report. *arXiv preprint arXiv:2512.16144*, 2025.

[117] T. Putterman, D. Lim, Y. Gelberg, M. M. Bronstein, S. Jegelka, and H. Maron. GL equivariant metanetworks for learning on low rank weight spaces. In *Learning on Graphs Conference (LoG)*, 2025.

[118] X. Qi, M. Chen, J. Ye, Y. He, and R. Xiao. Delving into Muon and beyond: Deep analysis and extensions. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

[119] Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[120] Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

[121] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018.

[122] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019.

[123] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

[124] A. Riabinin, E. Shulgin, K. Gruntkowska, and P. Richtárik. From Muon to Gluon: Bridging theory and practice of LMO-based optimizers for LLMs. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

[125] S. Schubert, P. Neubert, J. Pöschmann, and P. Protzel. Circular convolutional neural networks for panoramic images and laser data. In *IEEE Intelligent Vehicles Symposium (IV)*, 2019.

[126] A. Semenov, M. Pagliardini, and M. Jaggi. Benchmarking optimizers for large language model pretraining. *arXiv preprint arXiv:2509.01440*, 2025.

[127] N. Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[128] N. Shazeer and M. Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[129] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*, 2017.

[130] W. Shen, R. Huang, M. Huang, C. Shen, and J. Zhang. On the convergence analysis of Muon. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856.

[131] Y. Shen, Z. Guo, T. Cai, and Z. Qin. JetMoE: Reaching Llama2 performance with 0.1M dollars. *arXiv preprint arXiv:2404.07413*, 2024.

[132] H.-J. M. Shi, T.-H. Lee, S. Iwasaki, J. Gallego-Posada, Z. Li, K. Rangadurai, D. Mudigere, and M. Rabbat. A distributed data-parallel PyTorch implementation of the distributed Shampoo optimizer for training neural networks at-scale. *arXiv preprint arXiv:2309.06497*, 2023.

[133] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[134] C. Si, D. Zhang, and W. Shen. AdaMuon: Adaptive Muon optimizer. *arXiv preprint arXiv:2507.11005*, 2025.

[135] V. Singh, L. Krauss, S. Jaghouar, M. Sirovatka, C. Goddard, F. Obied, J. M. Ong, J. Straube, Fern, A. Harley, C. Stewart, C. Kealty, M. Panahi, S. Kirsten, A. Deshpande, A. Vij, A. Bresnu, P. Veldurthi, R. Ravishankar, H. Bishnoi, DatologyAI Team, Arcee AI Team, Prime Intellect Team, M. McQuade, J. Hagemann, and L. Atkins. Arcee Trinity Large technical report. *arXiv preprint arXiv:2602.17004*, 2026.

[136] StepFun Team, A. Huang, A. Li, A. Kong, B. Wang, B. Jiao, B. Dong, B. Wang, B. Chen, B. Li, B. Ma, C. Su, C. Miao, C. Wan, C. Lou, C. Hu, C. Xu, C. Yu, C. Feng, C. Yao, C. Han, D. Ma, D. Shi, D. Jiang, D. Ma, D. Sun, D. Qi, E. Liu, F. Zhang, F. Wan, G. Huang, G. Yan, G. Cao, G. Li, H. Cheng, H. Guo, H. Zhang, H. Nie, H. Jia, H. Lv, H. Zhou, H. Lv, H. Wang, H.-Y. Shum, H. Huang, H. Peng, H. Zhou, H. Wang, H. Chen, H. Zhu, H. Wu, H. Guo, J. Wang, J. Zhou, J. Sun, J. Wu, J. Zhang, J. Lv, J. Liu, J. Fu, J. Liu, J. Cheng, J. Luo, J. Yang, J. Zhou, J. Hou, J. Bai, J. Hu, J. Xie, J. Wu, J. Zhang, J. Zhou, J. Liu, J. Lin, K. M. Lo, K. Liang, K. Liu, K. Tan, K. Yan, K. Li, K. An, K. Lin, L. Yang, L. Lv, L. Zhao, L. Chen, L. Shi, L. Tan, L. Lin, L. Chen, L. Ma, M. Ren, M. Li, M. Li, M. Li, M. Zhang, M. Chen, M. Huang, N. Wang, P. Liu, Q. Han, Q. Zhao, Q. He, Q. Du, Q. Wu, Q. Sun, R. Yang, R. Miao, R. Han, R. Wan, R. Guo, S. Wang, S. Pang, S. Yang, S. Fan, S. Shang, S. Yang, S. Li, S. Tian, S. Liu, S. Wu, S. Chen, S. Yuan, T. Cao, T. Yue, T. Cheng, T. Li, T. Luo, W. You, W. Ji, W. Yuan, W. Zhang, W. Wu, W. Xie, W. Sun, W. Deng, W. Zheng, W. Xie, X. Wang, X. Kong, X. Liu, X. Zhang, X. Yang, X. Liu, X. Yuan, X. Jiao, X. Ren, X. Zhang, X. Li, X. Liu, X. Wu, X. Chen, X. Yang, X. Wang, X. Zhao, X. He, X. Feng, X. Cai, X. Zhou, Y. Yu, Y. Li, Y. Xu, Y. Lai, Y. Xu, Y. Wang, Y. Shen, Y. Zhu, Y. Lv, Y. Cao, Y. Gong, Y. Yang, Y. Yang, Y. Zhao, Y. Zhao, Y. Zhang, Y. Zhang, Y. Zhang, Y. Chen, Y. Zhao, Y. Long, Y. Wang, Y. Guan, Y. Zhou, Y. Peng, Y. Ding, Y. Fan, Y. Lu, Y. Yang, Y. Luo, Y. Zhao, Y. Peng, Y. Lin, Y. Lu, Y. Zhao, Y. Ju, Y. Zhang, Y. Li, Y. Yang, Y. Chen, Y. Cai, Z. Weng, Z. Hong, Z. Li, Z. Xie, Z. Ge, Z. Gong, Z. Zeng, Z. Lu, Z. Huang, Z. Chang, Z. Huang, Z. Hu, Z. Yang, Z. Wang, Z. Ren, Z. Zhang, and Z. Wang. Step 3.5 Flash: Open frontier-level intelligence with 11B active parameters. *arXiv preprint arXiv:2602.10604*, 2026.

[137] D. Su, A. Gu, J. Xu, Y. Tian, and J. Zhao. GaLore 2: Large-scale LLM pre-training by gradient low-rank projection. *arXiv preprint arXiv:2504.20437*, 2025.

[138] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063.

[139] W. Su. Isotropic curvature model for understanding deep learning optimization: Is gradient orthogonalization optimal? *arXiv preprint arXiv:2511.00674*, 2025.

[140] D. Sun and J. Sun. Löwner's operator and spectral functions in Euclidean Jordan algebras. *Mathematics of Operations Research*, 33(2):421–445, 2008.

[141] T. Tieleman and G. Hinton. Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude. Coursera: Neural Networks for Machine Learning, 2012.

[142] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[143] A. Veprikov, A. Bolatov, S. Horváth, A. Beznosikov, M. Takáč, and S. Hanzely. Preconditioned norms: A unified framework for steepest descent, quasi-Newton and adaptive methods. *arXiv preprint arXiv:2510.10777*, 2025.

[144] N. Vyas, D. Morwani, R. Zhao, I. Shapira, D. Brandfonbrener, L. Janson, and S. Kakade. SOAP: Improving and stabilizing Shampoo using Adam. In *International Conference on Learning Representations (ICLR)*, 2025.

[145] K. Wen, D. Hall, T. Ma, and P. Liang. Fantastic pretraining optimizers and where to find them. In *International Conference on Learning Representations (ICLR)*, 2026.

[146] M. Wortsman, T. Dettmers, L. Zettlemoyer, A. S. Morcos, A. Farhadi, and L. Schmidt. Stable and low-precision training for large-scale vision-language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[147] M. Wortsman, P. J. Liu, L. Xiao, K. Everett, A. Alemi, B. Adlam, J. D. Co-Reyes, I. Gur, A. Kumar, R. Novak, J. Pennington, J. Sohl-dickstein, K. Xu, J. Lee, J. Gilmer, and S. Kornblith. Small-scale proxies for large-scale transformer training instabilities. In *International Conference on Learning Representations (ICLR)*, 2024.

[148] S. Xie, T. Wang, S. Reddi, S. Kumar, and Z. Li. Structured preconditioners in adaptive optimization: A unified analysis. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025.

[149] T. Xie, H. Luo, H. Tang, Y. Hu, J. K. Liu, Q. Ren, Y. Wang, W. X. Zhao, R. Yan, B. Su, C. Luo, and B. Guo. Controlled LLM training on spectral sphere. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

[150] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T. Liu. On layer normalization in the transformer architecture. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

[151] C. Xu, W. Yan, and Y.-J. A. Zhang. FISMO: Fisher-structured momentum-orthogonalized optimizer. *arXiv preprint arXiv:2601.21750*, 2026.

[152] R. Xu, J. Li, and Y. Lu. On the width scaling of neural optimizers under matrix operator norms I: Row/column normalization and hyperparameter transfer. *arXiv preprint arXiv:2603.09952*, 2026.

[153] F. Xue, Z. Zheng, Y. Fu, J. Ni, Z. Zheng, W. Zhou, and Y. You. OpenMoE: An early effort on open Mixture-of-Experts language models. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

[154] Z. Yan, H. Bai, X. Yao, D. Liu, T. Liu, H. Liu, P. Li, E. Wu, S. Fan, L. Tao, R. Zhang, Y. Wang, S. Xu, J. Chang, X. Chen, K. Li, Y. Bai, G. Deng, N. Zheng, V. A. Korthikanti, A. Khattar, E. He, S. Govande, S. Lym, Z. Zhu, Q. Zhang, H. Yuan, X. Ren, D. Fu, T. Ma, S. Zhang, J. Shao, R. Wang, V. Rengasamy, R. Garg, S. Bhavani, X. Li, C. Zhou, D. Wu, Y. Wei, A. Aithal, M. Andersch, M. Shoeybi, J. Yao, and J. Yang. Scalable training of Mixture-of-Experts models with Megatron Core. *arXiv preprint arXiv:2603.07685*, 2026.

[155] G. Yang, E. J. Hu, I. Babuschkin, S. Sidor, X. Liu, D. Farhi, N. Ryder, J. Pachocki, W. Chen, and J. Gao. Tuning large neural networks via zero-shot hyperparameter transfer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[156] K. Yang and L. Lai. Manifold constrained steepest descent. *arXiv preprint arXiv:2601.21487*, 2026.

[157] S. Yang, Z. Wang, O. Balabanov, N. B. Erichson, and M. W. Mahoney. PRISM: Distribution-free adaptive computation of matrix functions for accelerating neural network training. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

[158] Y. You, I. Gitman, and B. Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

[159] Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, and C.-J. Hsieh. Large batch optimization for deep learning: Training BERT in 76 minutes. In *International Conference on Learning Representations (ICLR)*, 2020.

[160] H. Yuan, Y. Liu, S. Wu, X. Zhou, and Q. Gu. MARS: Unleashing the power of variance reduction for training large models. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025.

[161] J. Yuan, J. Zou, S. Wang, Y. Liu, and F. Nie. Nora: Normalized orthogonal row alignment for scalable matrix optimizer. *arXiv preprint arXiv:2605.03769*, 2026.

[162] J. Zeng, T. T.-K. Lau, S. Lin, and Y. Yao. Global convergence of block coordinate descent in deep learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

[163] B. Zhang and R. Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[164] J. Zhang, N. Amsel, B. Chen, and T. Dao. Gram Newton-Schulz, 2026. URL https://dao-ailab.github.io/blog/2026/gram-newton-schulz/.

[165] M. Zhang, Y. Liu, and H. Schaeffer. AdaGrad meets Muon: Adaptive stepsizes for orthogonal updates. *arXiv preprint arXiv:2509.02981*, 2025.

[166] R. Zhang, Y. Zhao, Z. Liu, Z. Wang, and Z. Zhang. Muon+: Towards better Muon via one additional normalization step. *arXiv preprint arXiv:2602.21545*, 2026.

[167] Y. Zhang, S. Xing, J. Huang, K. Lv, Y. Zhou, X. Qiu, Q. Guo, and K. Chen. Mousse: Rectifying the geometry of Muon with curvature-aware preconditioning. *arXiv preprint arXiv:2603.09697*, 2026.

[168] B. Zhao, N. Dehmamy, R. Walters, and R. Yu. Symmetry teleportation for accelerated optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[169] B. Zhao, R. M. Gower, R. Walters, and R. Yu. Improving convergence and generalization using parameter symmetries. In *International Conference on Learning Representations (ICLR)*, 2024.

[170] B. Zhao, R. Walters, and R. Yu. Symmetry in neural network parameter spaces. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856.

[171] J. Zhao, Z. Zhang, B. Chen, Z. Wang, A. Anandkumar, and Y. Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

[172] Y. Zhao, A. Gu, R. Varma, L. Luo, C.-C. Huang, M. Xu, L. Wright, H. Shojanazeri, M. Ott, S. Shleifer, A. Desmaison, C. Balioglu, P. Damania, B. Nguyen, G. Chauhan, Y. Hao, A. Mathews, and S. Li. PyTorch FSDP: Experiences on scaling fully sharded data parallel. *Proceedings of the VLDB Endowment*, 16(12): 3848–3860, 2023. doi: 10.14778/3611540.3611569.

[173] B. Zoph, I. Bello, S. Kumar, N. Du, Y. Huang, J. Dean, N. Shazeer, and W. Fedus. ST-MoE: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

Appendix

Contents

1 Introduction **1**

2 Equivariant Optimizers from Layerwise Symmetry **3**

2.1 A General Symmetry-Induced Optimizer Geometry 4

2.2 Bi-Orthogonal Equivariance for Ordinary Matrix Layers 4

3 Spectral Optimizers as the Bi-Orthogonally Equivariant Class **5**

4 General Symmetry-Compatible Optimizers via Equivariance **6**

4.1 Optimizers for Embeddings and LM Heads via LPRO Equivariance 6

4.1.1 Right-Spectral Optimizers 7

4.1.2 Row-Norm and Hybrid LPRO-Equivariant Optimizers 8

5 Numerical Experiments on Downsized gpt-oss Pre-Training **9**

6 Concluding Remarks **10**

References **11**

Appendix **22**

A Notation **24**

B Related Work **25**

B.1 Matrix-Gradient Optimizers 25

B.2 Matrix Optimization Problems, Löwner Operators, and Spectral Operators 26

B.3 Symmetry and Equivariance in Deep Learning 26

B.4 Non-Euclidean Norm-Based Steepest Descent and LMO-Based Frameworks 27

B.5 Modular Norm Theory 27

B.6 Rotation-Based Optimizers 27

C Supplemental Technical Background **28**

C.1 Matrix Analysis 28

D Geometric Misalignment of Coordinate-wise Adaptive Gradient Methods **28**

E Examples of Spectral and Equivariant Matrix Optimizers **30**

F Optimizers for SwiGLU MLP Projections **30**

G Optimizers for MoE Routers **31**

H A General and Unifying Symmetry-Induced Optimizer Geometry **34**
H.1 Symmetry-to-Optimizer Principle and Architecture–Optimizer Co-Design 36

I Proofs of Main Text **36**

J Implementation Details of Practical Optimizers **38**
J.1 Practical Optimizers for Embeddings, LM Heads, and MoE Routers 38
J.1.1 One-Sided Spectral Optimizers 38
J.1.2 Row-Norm-Based and Hybrid Variants 39
J.2 Numerical Algorithm for Matrix Inverse Square Root via Polar Express 41

K Additional Numerical Experiments **41**
K.1 Gemma 3 1B-Style Pre-Training 41
K.1.1 Gemma 3 1B-Style Pre-Training Learning Rate Sweep 42
K.1.2 Gemma 3 1B-Style Pre-Training Across Random Seeds 43
K.1.3 Vocabulary-Logit Growth and Final Logit Soft-Capping 44
K.1.4 Separation of the Embedding and LM-Head Contributions 46
K.2 Qwen3-0.6B-Style Pre-Training 48
K.3 OLMoE-1B-7B-Style Pre-Training 50
K.4 Load-Balancing Loss and Router z-Loss Diagnostics for Downsized gpt-oss Pre-Training 50
K.5 Cross-Model Comparison 56

L Experimental Details **57**
L.1 Gemma 3 1B-Style Pre-Training 58
L.2 Qwen3-0.6B-Style Pre-Training 59
L.3 OLMoE-1B-7B-Style Pre-Training 60
L.4 Downsized gpt-oss Pre-Training 61

M Prospect of Deep Learning Optimization **62**
M.1 Emergence of Geometry-Aware Layerwise Optimizers and Architecture–Optimizer Co-Design 62
M.2 Training Stabilization Tricks and Heuristics 62
M.3 Computational Challenges 63
M.4 Scalability Challenges 63

N Limitations **63**

Organization. In the appendix, we provide discussion on notation (Section A), related work (Section B), and supplementary technical background materials (Section C). Furthermore, we illustrate the geometric misalignment of coordinate-wise adaptive gradient methods in Section D, and give detailed exposition on examples and near-examples of spectral optimizers in the deep learning optimization literature in Section E. We then derive optimizers for SwiGLU MLPs and MoE router weights in Sections F and G respectively, based on the same equivariance principle proposed in the main text. We summarize this principle in Section H, leading to a general and unifying symmetry-induced optimizer geometry. We then provide omitted proofs from the main text in Section I, as well as details of the implementation of practical optimizers Section J. We also provide additional numerical experiments on model pre-training in Section K and details of numerical experiments in Section L. Finally, we provide our speculations on the prospect of deep learning optimization in Section M and limitations of this work in Section N.

Figure 3 summarizes the two contrasting perspectives on deep learning optimization developed in this paper.

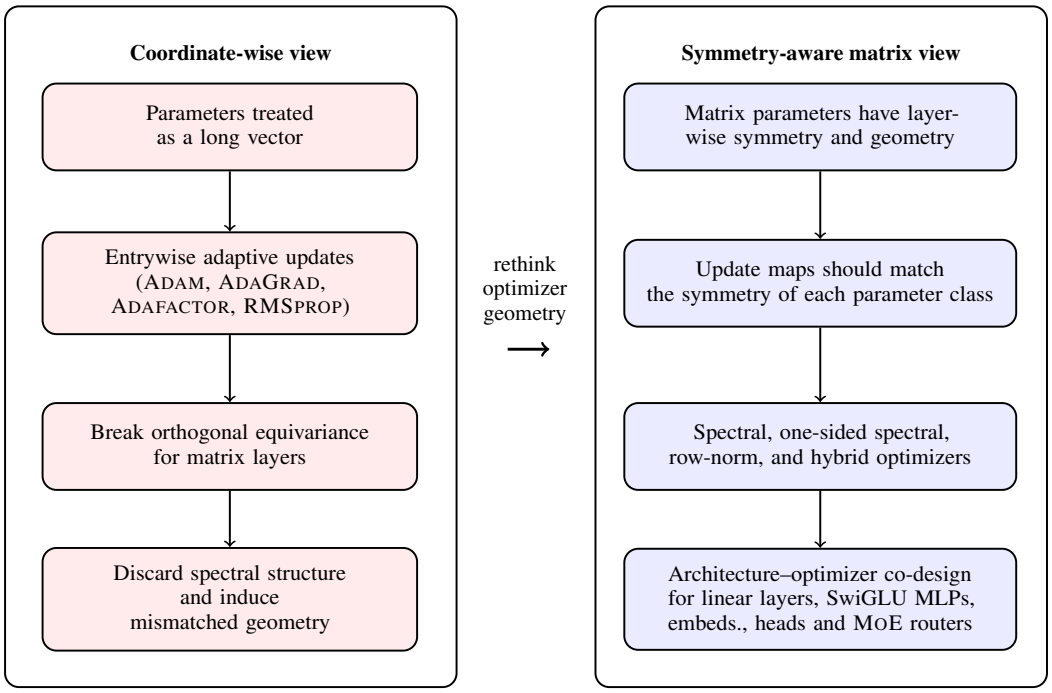


Figure 3: Two perspectives on deep learning optimization. Left: coordinate-wise adaptive methods treat matrix parameters as vectors and ignore matrix geometry. Right: the symmetry- and equivariance-based viewpoint developed in this paper leads to a family of equivariant, layer-specific optimizer classes and architecture–optimizer co-design.

A Notation

For any real-valued square matrix $S \in \mathbb{R}^{d\times d}$, $\mathrm{diag}(S) \in \mathbb{R}^d$ denotes the vector of its diagonal entries, $\mathrm{Diag}(S) \in \mathbb{R}^{d\times d}$ the diagonal matrix with diagonal entries equal to those of S, and $\mathrm{tr}(S)$ is its trace. For any $x \in \mathbb{R}^d$, $\mathrm{Diag}(x) \in \mathbb{R}^{d\times d}$ is the diagonal matrix with diagonal entries equal to the entries of x. We define $m \wedge n \coloneqq \min\{m, n\}$. For any $m \times n$ real-valued matrices $A \coloneqq (a_{i,j})_{1\leqslant i\leqslant m, 1\leqslant j\leqslant n}$ and $B \coloneqq (b_{i,j})_{1\leqslant i\leqslant m, 1\leqslant j\leqslant n}$, we denote the Frobenius inner product of A and B by $\langle\!\langle A, B\rangle\!\rangle_{\mathrm{F}} \coloneqq \mathrm{tr}(A^\top B) = \sum_{i,j} a_{i,j}b_{i,j}$. For a matrix $A \in \mathbb{R}^{m\times n}$, we denote its Frobenius norm by $|\!|\!|A|\!|\!|_{\mathrm{F}} \coloneqq \sqrt{\langle\!\langle A, A\rangle\!\rangle_{\mathrm{F}}}$, its spectral norm by $|\!|\!|A|\!|\!|_{\mathrm{S}} \coloneqq \sup_{x\in\mathbb{R}^n, x\neq 0}\{\|Ax\|_2/\|x\|_2\}$, its nuclear norm by $|\!|\!|A|\!|\!|_{\mathrm{nuc}} \coloneqq \sum_{i=1}^{m\wedge n} \sigma_i(A)$, where $\sigma(A) = (\sigma_1(A), \ldots, \sigma_{m\wedge n}(A))^\top$ is the vector of nonincreasing ordered singular values of A, and its max norm by $|\!|\!|A|\!|\!|_{\max} \coloneqq \max_{1\leqslant i\leqslant m, 1\leqslant j\leqslant n} |a_{i,j}|$. The Schatten p-norm of A is denoted by $|\!|\!|A|\!|\!|_p \coloneqq \|\sigma(A)\|_p$. The Hadamard product of $A \in \mathbb{R}^{m\times n}$

and $B \in \mathbb{R}^{m\times n}$ is denoted by $A \odot B \coloneqq (a_{i,j}b_{i,j})_{1\leqslant i\leqslant m, 1\leqslant j\leqslant n}$. For the the matrix $A \in \mathbb{R}^{m\times n}$, we denote by $\mathrm{vec}(A) \in \mathbb{R}^{mn}$ its vectorization by rows. Conversely, for $x \in \mathbb{R}^{mn}$, we write $\mathrm{reshape}(x, m, n) \in \mathbb{R}^{m\times n}$ for the inverse operation, so that $\mathrm{reshape}(\mathrm{vec}(A), m, n) = A$ for all $A \in \mathbb{R}^{m\times n}$. Let $\mathbb{S}^d \coloneqq \{A \in \mathbb{R}^{d\times d} : A = A^\top\}$ denote the space of real symmetric matrices in $\mathbb{R}^{d\times d}$, $\mathbb{S}^d_+ \coloneqq \{A \in \mathbb{S}^d : A \succcurlyeq 0\}$ the set of symmetric positive semidefinite matrices, and $\mathbb{S}^d_{++} \coloneqq \{A \in \mathbb{S}^d : A \succ 0\}$ the set of symmetric positive definite matrices, where $\succcurlyeq$ and $\succ$ denote Löwner orders. Let $\mathbb{O}^d \coloneqq \{A \in \mathbb{R}^{d\times d} : A^\top A = AA^\top = I_d\}$ denote the set of $d \times d$ orthogonal matrices, where $I_d \in \mathbb{R}^{d\times d}$ is the $d \times d$ identity matrix. Let $\mathbb{P}^d \coloneqq \{P \in \{0,1\}^{d\times d} : P\mathbf{1}_d = \mathbf{1}_d,\ P^\top\mathbf{1}_d = \mathbf{1}_d\}$ denote the set of $d \times d$ permutation matrices, where $\mathbf{1}_d$ is the all-ones vector in $\mathbb{R}^d$. Let $\mathcal{E}$ be a Euclidean space endowed with an inner product $\langle\cdot,\cdot\rangle$ and the induced norm $\|\cdot\|$. The domain of a function $f\colon \mathcal{E} \to \overline{\mathbb{R}} \coloneqq \mathbb{R} \cup \{\pm\infty\}$ is $\mathrm{dom}\, f \coloneqq \{x \in \mathcal{E} : f(x) < \infty\}$. A function $f\colon \mathcal{E} \to \overline{\mathbb{R}}$ is said to be *proper* if it has a nonempty domain. The (convex) indicator function $\iota_{\mathcal{C}}(x)$ of a nonempty closed convex set $\mathcal{C}$ at x equals 0 if $x \in \mathcal{C}$ and $+\infty$ otherwise. The Euclidean projection of x onto a nonempty closed convex set $\mathcal{C}$ is denoted by $\mathrm{proj}_{\mathcal{C}}(x)$. $\mathbb{N}$ denotes the set of nonnegative integers and $\mathbb{N}^* \coloneqq \mathbb{N} \setminus \{0\}$ denotes the set of positive integers. For a function $f\colon \mathcal{E} \to \overline{\mathbb{R}}$, we use $\mathrm{argmin}\, f$ to denote the unique minimizer of f.

B Related Work

We discuss closely related work and highlight connections between our framework and existing paradigms for designing matrix-gradient optimizers in modern deep learning.

B.1 Matrix-Gradient Optimizers

The recent release of MUON [73], together with its strong empirical performance in the `modded-nanogpt` speedrun [72], has renewed interest in matrix-gradient optimizers for deep learning. This has led to a rapidly growing line of work on geometry-aware and matrix-structured optimization methods [25, 27, 38, 45, 53, 68, 70, 80, 83, 113, 118, 134, 143, 151, 152, 165, 167]. Conceptually, MUON is closely related to stochastic spectral descent (SSD) [20–22], since both methods can be derived from steepest descent with respect to the spectral norm.

On the theoretical side, local one-step analyses of simplified MUON-type updates have been developed in [29, 54, 139]. Several recent works also study convergence rates and optimization guarantees for MUON under different assumptions [23, 75, 79, 83, 90, 104, 130]. Our work is aligned with this broad effort, but differs in emphasis: rather than viewing matrix-gradient optimizers primarily as normalization or preconditioning heuristics, we derive them from symmetry and equivariance principles for matrix-valued neural network parameters.

A separate but related line of work develops matrix-gradient optimizers from second-order or preconditioning perspectives. These include Kronecker-factored or layerwise preconditioners such as K-FAC [40, 105], SHAMPOO [7, 41, 59, 132], BFGS and L-BFGS-type methods [51], SOAP [144], KL-Shampoo and KL-SOAP [97], and learned or adaptive preconditioners such as preconditioned SGD (PSGD) [91, 115]. These methods typically approximate curvature or preconditioning structure, whereas spectral and polar updates can also be understood as enforcing equivariance properties of the update map itself. This distinction is important for our framework, since the appropriate optimizer geometry depends on the symmetry group of the layer, not only on curvature approximation.

Another related direction studies constraints directly on the weights in addition to the update directions. In particular, recent works connect MUON-type updates to optimization on the Stiefel manifold [11, 19], leading to manifold constrained optimizers [57, 149, 156].

Finally, other optimization techniques have also been adapted to matrix-gradient settings, including variance reduction in MARS-M [99, 160] and low-rank gradient projection methods such as GaLore [137, 171]. These methods address computational or statistical efficiency through variance control or low-rank subspace restriction. We refer readers to the recent review [114] for a broader overview of geometry-aware optimization methods in deep learning.

B.2 Matrix Optimization Problems, Löwner Operators, and Spectral Operators

Matrix optimization problems (MOPs) have long been studied as a distinct class of optimization problems, rather than merely as vector optimization problems written in matrix form [35, 36]. A central reason is that matrices carry algebraic and geometric structures—including eigenvalues, singular values, invariant subspaces, ranks, and unitary symmetries—that are destroyed or obscured by vectorization. Seminal work in convex matrix analysis developed the foundations for understanding convex and unitarily invariant functions of matrices [85, 86], and established deep connections between matrix functions, eigenvalue optimization, and spectral optimization [87, 88].

A key concept in this literature is that of a *spectral function* [15, 63, 65]. For symmetric matrices, spectral functions depend on a matrix only through its eigenvalues; for rectangular matrices, unitarily invariant functions depend on a matrix only through its singular values. Such functions naturally give rise to *spectral operators*: matrix-valued maps that preserve singular vectors and act only on singular values. In the rectangular case, a typical spectral operator, also known as a *Löwner operator* [140], has the form $G = U\,\mathrm{Diag}(\sigma(G))V^\top \mapsto \mathfrak{T}(G) = U\,\mathrm{Diag}(\psi(\sigma(G)))V^\top$, where ψ is a map acting on the singular values. This is precisely the operator-theoretic structure underlying many matrix-gradient optimizers, including stochastic spectral descent, MUON, and polar gradient methods.

Our work draws on this matrix-optimization perspective, but applies it to modern neural network training. We view the gradient of a matrix-valued parameter not as an arbitrary vector in $\mathbb{R}^{mn}$, but as a matrix whose row, column, and singular-vector geometries should be respected by the optimizer. This leads naturally to spectral and one-sided spectral update rules. In particular, the orthogonal polar factor $\mathrm{polar}(G) = UV^\top$ is a Löwner operator obtained by replacing all nonzero singular values of G by one. More generally, row-norm, right-spectral, left-spectral, and hybrid updates can be understood as optimizer-specific matrix operators chosen to match the symmetry group of the corresponding neural network layer.

This perspective distinguishes our work from approaches that treat matrix parameters only through vectorized coordinates or through a fixed choice of norm. Instead, the relevant matrix operator is determined by the symmetry and geometry of the parameter class: full spectral operators for standard linear layers, right-spectral or row-norm operators for vocabulary-indexed embedding and head matrices, and left-spectral or centered row-wise operators for MoE router matrices.

B.3 Symmetry and Equivariance in Deep Learning

Symmetry properties of neural networks have recently emerged as an important organizing principle for understanding optimization, generalization, and representation learning. Many neural network architectures admit nontrivial group actions on their parameters or hidden representations, and the training dynamics induced by certain optimizers may be equivariant with respect to these actions. Various examples of equivariant optimization algorithms based on gradient descent (GD) and stochastic gradient descent (SGD) are discussed in [1]. For instance, SGD training of fully-connected networks with Gaussian initialization is orthogonally equivariant [110], while networks with residual connections exhibit natural permutation equivariances [60]. Similar equivariance properties can be established for convolutional networks with circular convolutions [125] and transformer architectures [142]. More generally, Abbe and Boix-Adsera [1] formalized equivariance properties of GD and SGD under suitable regularity conditions. Relatedly, Li et al. [92] studied how optimizer choices, including ADAGRAD and ADAM, interact with symmetry and reparameterization structure.

Our work differs from this line of research in focus and granularity. Existing works primarily study the equivariance properties of an entire network architecture under a fixed optimizer, such as GD, SGD, ADAGRAD, or ADAM. By contrast, we study the symmetry structure of individual matrix-valued layers and use these symmetries to derive compatible layerwise optimizer classes.

A related body of work studies how neural network symmetries affect the loss landscape, optimization trajectories, and learned representations [168–170]. These works highlight that symmetries are not merely formal properties of architectures, but can shape the behavior of training and the structure of learned solutions. Our framework complements this perspective by turning symmetry considerations into concrete optimizer-design principles for matrix-valued parameters.

There are also connections to parameter-efficient fine-tuning methods such as LORA, whose low-rank factorization introduces additional gauge symmetries and non-identifiability in the factorized weights

[117]. Such settings further emphasize that optimizer behavior depends on the geometry of the chosen parameterization. While our main focus is full-parameter pre-training, the same symmetry-compatible viewpoint suggests that optimizers for low-rank-adapted or factorized layers should also be designed with their intrinsic invariances in mind.

B.4 Non-Euclidean Norm-Based Steepest Descent and LMO-Based Frameworks

A prominent line of recent work interprets MUON and related methods through the lens of non-Euclidean steepest descent, trust-region methods, and linear minimization oracle (LMO) frameworks [10, 12, 13, 73, 78, 113, 124]. In these views, MUON can be understood as a normalized steepest descent method with respect to a non-Euclidean matrix norm—most notably the spectral norm—and this perspective has also been used to connect MUON to optimizers such as SIGNSGD [14] and SHAMPOO [59]. Related work has further explored alternative norm choices and generalized preconditioned steepest descent constructions [27, 80, 143, 152].

While these frameworks provide useful algorithmic interpretations of MUON and its variants, they leave open a fundamental question: what is the *principled* criterion for choosing the "correct" norm for a given layer? In practice, existing prescriptions are not always fully aligned with actual usage. For example, Large et al. [81] suggest using the maximum column ℓ_2-norm for embedding layers, whereas in practice ADAMW is often used for embeddings while MUON is applied to other matrix parameters in `modded-nanogpt` speedrunning [72]. More broadly, several works suggest that one may use various $\ell_p \to \ell_q$ operator norms for steepest descent or LMO for embedding and LM head layers (see e.g., [113]). Our theoretical development suggests that such norm choices might generally not be geometrically principled for matrix-valued parameters.

In contrast, our symmetry-based analysis provides a concrete design principle. For matrix parameters whose geometry should respect orthogonal changes of basis, only unitarily invariant matrix norms are appropriate, since they are precisely the norms compatible with orthogonal invariance. By comparison, the $\ell_p \to \ell_q$ operator norm is generally *not* unitarily invariant, except in the special case $p = q = 2$, which reduces to the spectral norm. Accordingly, non-unitarily invariant norms typically induce an incorrect matrix geometry. This offers a possible explanation for the gap between certain norm-based theoretical prescriptions and empirical practice.

B.5 Modular Norm Theory

Our right-spectral viewpoint is closely related in spirit to modular norm theory [13, 81]: both seek architecture-aware optimizer geometries derived from structural properties of the module rather than from an ambient coordinate system. The two approaches are nevertheless conceptually distinct.

Modular norm theory derives updates from steepest descent under module-adapted operator norms, and is primarily motivated by scale transfer across width and depth through recursively constructed global norms. By contrast, right-spectral optimizers are derived directly from left-permutation and right-orthogonal equivariance, leading to update rules of the form (3), namely spectral transformations determined by the right Gram matrix $D^\top D$. In this sense, right-spectral optimizers form a symmetry-derived subclass of LPRO-equivariant updates, whereas modular-norm-based methods may also include coordinate-dependent row-wise or column-wise transformations that are compatible with only part of the underlying symmetry.

From this perspective, the two frameworks may be viewed as complementary. Modular norm theory aims to provide the correct *scale invariance*, namely how learning rates and normalized updates should behave as architecture width and depth vary. Our spectral framework instead aims to provide the correct *directional geometry*, namely how update directions should respect the invariance structure of different layer types. Their combination suggests the possibility of a fully geometry-aware optimizer that uses modular-norm-based global scaling together with layerwise spectral, right-spectral, or left-spectral update directions dictated by symmetry.

B.6 Rotation-Based Optimizers

The recent work [53] is perhaps the closest in spirit to our approach, but there are several important differences. First, we do not assume that the objective function itself is rotationally, or more generally bi-orthogonally, invariant. Rather, we derive optimizer classes from the symmetry structure of the

parameter geometry and the corresponding equivariance properties of update rules. This distinction matters, since exact rotational invariance of the objective may be difficult to verify and can fail in practice.

Second, we do not advocate a single update rule for all matrix-valued parameters. Instead, our framework emphasizes that different layer types admit different symmetry groups and therefore should, in principle, use different optimizer geometries. In particular, embedding and LM head matrices possess different equivariance properties from weight matrices in linear and attention layers, which leads naturally to an architecture–optimizer co-design perspective.

Third, the update rules in ARO for matrix parameters are based only on left multiplication by a rotation matrix. By contrast, our framework includes full spectral, right-spectral, left-spectral, row-norm-based, and hybrid classes, depending on the relevant symmetry structure of the layer. In this sense, our theory provides a broader symmetry-based taxonomy of matrix-gradient optimizers.

C Supplemental Technical Background

We include several technical definitions arising in matrix analysis, which is closely related to the derivation of spectral optimizers via steepest descent. Details of these materials can be found in [15, 65, 66, 88].

C.1 Matrix Analysis

Let us recall the notion of unitarily invariant norms, which provide the norm-level analogue of left-right orthogonal symmetry. Since we work over the field of real numbers $\mathbb{R}$, "unitarily invariant" here is equivalent to invariance under left and right orthogonal transformations.

Definition C.1 (Unitarily invariant norms)**.** A norm $\|\cdot\|\colon \mathbb{R}^{m\times n} \to \mathbb{R}_+$ is said to be *unitarily invariant* if$\|A\| = \|UAV\|$ for all $A \in \mathbb{R}^{m\times n}$ and all orthogonal matrices $U \in \mathbb{O}^m$ and $V \in \mathbb{O}^n$. A *unitarily invariant matrix norm* is a unitarily invariant norm on $\mathbb{R}^{m\times n}$ that is also submultiplicative.

Unitarily invariant norms can be completely characterized through symmetric gauge functions acting on singular values.

Definition C.2 (Symmetric gauge functions)**.** A function $\psi\colon \mathbb{R}^r \to \mathbb{R}_+$ is a *symmetric gauge function* if: (i) ψ is a norm; (ii) $\psi(|x|) = \psi(x)$ for all $x \in \mathbb{R}^r$, where $|\cdot|$ is understood coordinatewise; and (iii) $\psi(Px) = \psi(x)$ for all permutation matrices $P \in \mathbb{P}^r$ and all $x \in \mathbb{R}^r$.

Proposition C.1. *A norm $\|\cdot\|\colon \mathbb{R}^{m\times n} \to \mathbb{R}_+$ is unitarily invariant if and only if there exists a symmetric gauge function ψ such that $\|W\| = \psi(\sigma(W)) = (\psi \circ \sigma)(W)$, where $\sigma(W)$ is the vector of singular values of W arranged in descending order.*

Example C.1. Important examples of unitarily invariant matrix norms include the Schatten p-norms $|||\cdot|||_p$ for $p \in [1, \infty]$, including the nuclear norm $|||\cdot|||_{\mathrm{nuc}}$, Frobenius norm $|||\cdot|||_{\mathrm{F}}$, and the spectral norm $|||\cdot|||_{\mathrm{S}}$. Another notable example is the Ky Fan k-norm.

D Geometric Misalignment of Coordinate-wise Adaptive Gradient Methods

If an optimizer for matrix parameters in linear layers does not respect the intrinsic geometry of the parameter space through bi-orthogonal equivariance, several issues arise. In particular, for coordinate-wise adaptive gradient methods, the optimizer iterates depend on arbitrary coordinate choices: rotating the input space can change the optimizer itself, leading to different training dynamics under equivalent reparameterizations.

For a vector parameter $w \in \mathbb{R}^n$, the general form of a coordinate-wise adaptive gradient method is

$$(\forall k \in \mathbb{N}) \qquad w_{k+1} = w_k - \gamma_k\, \mathfrak{U}(d_k),$$

where d_k is an update direction and $\mathfrak{U}\colon \mathbb{R}^n \to \mathbb{R}^n$ is a vector-valued map. A simple but important observation is that optimizers built only from linear operations, such as additions, subtractions, scalar multiplications, and their linear combinations, behave analogously for vector and matrix parameters. By contrast, coordinate-wise adaptive methods such as ADAM rely on elementwise nonlinear operations, including division, squaring, and square roots. These operations distort the

geometry of the original update direction, whether it is formed from the gradient itself or from a momentum term.

While spectral methods (e.g., spectral descent and MUON; formally introduced in Section 3) respect bi-orthogonal equivariance, coordinate-wise methods are typically equivariant only under the much smaller signed permutation group. As a result, they fail to respect the intrinsic matrix geometry of the optimization problem. This mismatch is especially problematic in deep learning, where the optimization landscape often exhibits an intrinsically low-dimensional structure. Spectral optimizers are designed to adapt to this geometry, whereas coordinate-wise methods largely ignore it. Moreover, as model size increases, the advantage of such geometry-aware methods typically becomes more pronounced.

Sign descent. For a matrix gradient G, sign descent or SIGNSGD [14] uses the coordinate-wise signs $\mathrm{sgn}(G)$ of the gradient as the update direction, which satisfies $\langle\!\langle G, \mathrm{sgn}(G)\rangle\!\rangle_{\mathrm{F}} = \|G\|_1$. Thus, sign descent is governed by the duality associated with the coordinate-wise ℓ_∞-norm, rather than by any unitarily invariant matrix geometry (see Definition C.1).

ADAM. More generally, ADAM [77] can be viewed as a smoothed version of sign descent, since its update is dominated by coordinate-wise normalization. In particular, ADAM applies the coordinate-wise scaling (omitting bias corrections for simplicity)

$$W_{ij} \leftarrow W_{ij} - \gamma \cdot \frac{M_{ij}}{\sqrt{V_{ij}} + \varepsilon},$$

where M_{ij} and V_{ij} denote the first- and second-moment statistics at coordinate (i, j). This update does not transform equivariantly under bi-orthogonal reparameterizations $W \mapsto \widetilde{W} := PWQ^\top$ for $P \in \mathbb{O}^m$ and $Q \in \mathbb{O}^n$. Consequently, the update direction of ADAM depends on how the entries of W are indexed, rather than only on the intrinsic geometry of W as a matrix.

Furthermore, coordinate-wise methods tend to inject high-rank coordinate noise even when the underlying gradient G is low rank. In contrast, spectral methods act directly on the matrix structure of G, and therefore remain aligned with the low-dimensional geometry that frequently governs deep network optimization.

Sign descent as a special case of spectral descent. Following similar arguments in [83], we further interpret sign descent as spectral descent applied to the diagonal matrization of the vectorized matrix parameter. Define the *diagonal matrization of the vectorization* of G by $\widetilde{G} := \mathrm{Diag}(\mathrm{vec}(G)) \in \mathbb{R}^{mn\times mn}$. The following lemma makes this connection precise.

Lemma D.1. *Let $G := \nabla_W f(W) \in \mathbb{R}^{m\times n}$ contain no exact zero entries, and define $\widetilde{G} := \mathrm{Diag}(\mathrm{vec}(G)) \in \mathbb{R}^{mn\times mn}$. Then the orthogonal polar factor of $\widetilde{G}$ is $\mathrm{polar}(\widetilde{G}) = \mathrm{Diag}(\mathrm{sgn}(\mathrm{vec}(G)))$. Consequently,*

$$\mathrm{sgn}(G) = \mathrm{reshape}(\mathrm{diag}(\mathrm{polar}(\widetilde{G})), m, n).$$

Proof. Since $\widetilde{G} = \mathrm{Diag}(\mathrm{vec}(G))$ is diagonal, its singular values are given by the absolute values of its diagonal entries. Hence we may write its singular value decomposition as

$$\widetilde{G} = I_{mn}\ \mathrm{Diag}(|\mathrm{vec}(G)|)\ \mathrm{Diag}(\mathrm{sgn}(\mathrm{vec}(G))),$$

or equivalently,

$$\widetilde{G} = \mathrm{Diag}(\mathrm{sgn}(\mathrm{vec}(G)))\ \mathrm{Diag}(|\mathrm{vec}(G)|).$$

Therefore, by the definition of the orthogonal polar factor, we have $\mathrm{polar}(\widetilde{G}) = \mathrm{Diag}(\mathrm{sgn}(\mathrm{vec}(G)))$. Taking the diagonal of $\mathrm{polar}(\widetilde{G})$ and reshaping it back to an $m \times n$ matrix yields

$$\mathrm{reshape}(\mathrm{diag}(\mathrm{polar}(\widetilde{G})), m, n) = \mathrm{reshape}(\mathrm{sgn}(\mathrm{vec}(G)), m, n) = \mathrm{sgn}(G).$$

□

In other words, coordinate-wise sign descent can be viewed as spectral descent applied to the highly degenerate diagonal lifting $\widetilde{G} = \mathrm{Diag}(\mathrm{vec}(G)) \in \mathbb{R}^{mn\times mn}$. Thus, sign-based methods inherit only the trivial geometry of this pathological diagonal representation, rather than the intrinsic matrix geometry of G itself. This highlights a fundamental geometric mismatch in coordinate-wise updates, which is further worsened when the parameter dimensions m and n and hence model sizes grow.

E Examples of Spectral and Equivariant Matrix Optimizers

The simplest spectral optimizer is vanilla gradient descent, for which the spectral operator is the identity map. Its Polyak, Nesterov, and EMA-momentum variants remain spectral because bi-orthogonally equivariant momentum constructions composed with spectral operators preserve spectrality. Other examples include stochastic spectral descent [20, 22], MUON [73], SCION [113], and POLARGRAD [83]. Power-type spectral maps $\psi(t) = t^p$, including $p \in \{1/2, 1/4\}$, are also studied in [118].

It is useful to distinguish spectral operators acting on the current update direction from history-dependent optimizers whose states evolve equivariantly. For example, SHAMPOO [7, 59] maintains left and right preconditioners based on moving averages of $G_k G_k^\top$ and $G_k^\top G_k$, and applies

$$L_k^{-1/4} G_k R_k^{-1/4}.$$

This update is bi-orthogonally equivariant when the state variables (L_k, R_k) transform accordingly, but it is not generally a spectral operator of the current gradient alone, since the preconditioners need not share the singular-vector basis of G_k. In special aligned cases, it reduces to a singular-value transformation and hence becomes spectral. One-sided SHAMPOO [89, 148] and ASGO [6] admit a similar interpretation.

SOAP [144] is also geometry-aware but generally not spectral in the strict direction-wise sense. It rotates gradients into learned eigenspaces and applies coordinate-wise adaptive scaling in those evolving coordinates. Thus, the update depends on history-dependent bases and coordinate-wise statistics, not only on the singular values of the current gradient. It becomes spectral only in special cases where the learned eigenspaces align with the singular-vector basis and the adaptive scaling acts symmetrically across singular directions.

Normalization preserves spectrality when the normalizing scalar is unitarily invariant. If $\mathcal{U}$ is spectral and $\alpha\colon \mathbb{R}^{m\times n} \to \mathbb{R}_+$ is a positive spectral scalar function, then $D \mapsto \mathcal{U}(D)/\alpha(D)$ is again spectral. In particular, normalization by a unitarily invariant matrix norm preserves spectrality. By contrast, row-wise or column-wise normalization generally breaks full bi-orthogonal equivariance because it depends on a preferred coordinate system. Such normalizations may still be appropriate for layers with one-sided or permutation symmetries, but they are not spectral operators for ordinary matrix layers.

More broadly, steepest descent for matrix-valued parameters is compatible with bi-orthogonal symmetry when the underlying norm is unitarily invariant. The unit ball of a unitarily invariant norm is invariant under $D \mapsto PDQ^\top$, so the associated steepest descent direction transforms equivariantly under left and right orthogonal changes of coordinates. Non-unitarily invariant norms, such as general induced $\ell_p \to \ell_q$ operator norms with $(p, q) \neq (2, 2)$, generally fail this property and impose a coordinate-dependent geometry on the update.

F Optimizers for SwiGLU MLP Projections

We next consider SwiGLU MLP projection matrices [28, 127]. Unlike ordinary linear and attention projection matrices, SwiGLU projections do not possess full bi-orthogonal symmetry. Instead, their natural symmetry is the permutation symmetry of intermediate neurons. This suggests that the gate and up projections should use left-permutation/right-orthogonal equivariant updates, while the down projection should use the corresponding transposed geometry. Concretely, for $W_{\text{gate}}, W_{\text{up}} \in \mathbb{R}^{d_{\text{ff}}\times d_{\text{model}}}$ and $W_{\text{down}} \in \mathbb{R}^{d_{\text{model}}\times d_{\text{ff}}}$, we apply the LPRO-compatible optimizer classes of **??** to W_{gate}, W_{up}, and $W_{\text{down}}^\top$.

This viewpoint is closely related to Aurora [34], an optimizer designed for the up and gate projections in SwiGLU MLPs. Aurora alternates between row-normalization and polar steps on the momentum, which is similar in spirit to the hybrid row-norm/right-spectral optimizers developed above. Our contribution here is to derive this type of update from the intermediate-neuron permutation symmetry of the SwiGLU block.

Proposition F.1 (Intermediate-neuron permutation symmetry of SwiGLU MLPs)**.** *Let us consider the SwiGLU MLP block defined by*

$$\mathrm{SwiGLU}(x; W_{\text{gate}}, W_{\text{up}}, W_{\text{down}}) \coloneqq W_{\text{down}}(\sigma(W_{\text{gate}}x) \odot (W_{\text{up}}x)),$$

where $W_{\mathrm{gate}}, W_{\mathrm{up}} \in \mathbb{R}^{d_{\mathrm{ff}} \times d_{\mathrm{model}}}$, $W_{\mathrm{down}} \in \mathbb{R}^{d_{\mathrm{model}} \times d_{\mathrm{ff}}}$, σ *is applied coordinatewise, and* $\odot$ *denotes the Hadamard product. Let* $P \in \mathbb{P}^{d_{\mathrm{ff}}}$ *be a permutation matrix, and define* $\widetilde{W}_{\mathrm{gate}} \coloneqq PW_{\mathrm{gate}}$, $\widetilde{W}_{\mathrm{up}} \coloneqq PW_{\mathrm{up}}$, *and* $\widetilde{W}_{\mathrm{down}} \coloneqq W_{\mathrm{down}}P^\top$. *Then, for every input* $x \in \mathbb{R}^{d_{\mathrm{model}}}$,

$$\mathrm{SwiGLU}(x; \widetilde{W}_{\mathrm{gate}}, \widetilde{W}_{\mathrm{up}}, \widetilde{W}_{\mathrm{down}}) = \mathrm{SwiGLU}(x; W_{\mathrm{gate}}, W_{\mathrm{up}}, W_{\mathrm{down}}).$$

Proof. Since σ is applied coordinatewise and P is a permutation matrix, $\sigma(PW_{\mathrm{gate}}x) = P\sigma(W_{\mathrm{gate}}x)$. Moreover, $(Pu) \odot (Pv) = P(u \odot v)$ for all $u, v \in \mathbb{R}^{d_{\mathrm{ff}}}$. Hence

$$\begin{aligned}\mathrm{SwiGLU}(x; \widetilde{W}_{\mathrm{gate}}, \widetilde{W}_{\mathrm{up}}, \widetilde{W}_{\mathrm{down}}) &= W_{\mathrm{down}}P^\top(\sigma(PW_{\mathrm{gate}}x) \odot (PW_{\mathrm{up}}x)) \\ &= W_{\mathrm{down}}P^\top P(\sigma(W_{\mathrm{gate}}x) \odot (W_{\mathrm{up}}x)) \\ &= W_{\mathrm{down}}(\sigma(W_{\mathrm{gate}}x) \odot (W_{\mathrm{up}}x)).\end{aligned}$$

□

Remark F.1 (Failure of full left-orthogonal symmetry). The permutation matrix P in Proposition F.1 cannot generally be replaced by an arbitrary orthogonal matrix $Q \in \mathbb{O}^{d_{\mathrm{ff}}}$. For an elementwise nonlinearity σ, one typically has $\sigma(Qz) \neq Q\sigma(z)$, and similarly $(Qa) \odot (Qb) \neq Q(a \odot b)$ for a general orthogonal matrix Q. Thus, the SwiGLU intermediate dimension has a discrete permutation symmetry, not a full left-orthogonal symmetry.

Proposition F.1 implies that optimizers for W_{gate} and W_{up} should commute with permutations of their rows and orthogonal changes of basis in the model dimension. Thus, the same LPRO-compatible row-norm, right-spectral, and hybrid row-norm/right-spectral updates developed for embeddings and LM heads apply directly to these matrices. For the down projection, the intermediate-neuron axis appears as the column dimension, so the same principle applies to $W_{\mathrm{down}}^\top$. Equivalently, one may view the down-projection update as a right-permutation/left-orthogonal analogue of the LPRO updates.

More generally, this intermediate-neuron permutation symmetry is not specific to SwiGLU. Any feed-forward block whose hidden nonlinearity is applied coordinatewise admits a permutation symmetry of the hidden units: simultaneously permuting the rows of the input projection and the corresponding columns of the output projection leaves the represented function unchanged. For gated MLPs such as GLU, GeGLU, ReGLU, and SwiGLU, the same symmetry holds provided the gate and value projections are permuted together, along with the corresponding columns of the down projection.

G Optimizers for MoE Routers

We now consider the router matrix in a mixture-of-experts (MoE) model. Unlike ordinary linear layers, embeddings, and LM heads, the router has an additional symmetry: experts are interchangeable, and the softmax is invariant under adding a shared scalar offset to all expert logits.

Let $W \in \mathbb{R}^{e \times d}$ denote the router matrix, where $e \in \mathbb{N}^*$ is the number of experts and $d \in \mathbb{N}^*$ is the hidden dimension. The routing distribution is $p(x; W) = \mathrm{softmax}(Wx)$ for $x \in \mathbb{R}^d$. Since permuting the rows of W only relabels the experts, and since $\mathrm{softmax}(z + c\mathbf{1}_e) = \mathrm{softmax}(z)$ for all $z \in \mathbb{R}^e$ and $c \in \mathbb{R}$, the router parameters admit the symmetry

$$(\forall P \in \mathbb{P}^e, \forall a \in \mathbb{R}^d) \qquad W \mapsto PW + \mathbf{1}_e a^\top.$$

Indeed,

$$(\forall x \in \mathbb{R}^d) \qquad (PW + \mathbf{1}_e a^\top)x = P(Wx) + (a^\top x)\mathbf{1}_e,$$

so the router logits are unchanged up to expert relabeling and a shared logit shift.

This symmetry suggests that router updates should be defined on the centered expert geometry. Let

$$\Pi_\perp \coloneqq I_e - \frac{1}{e}\mathbf{1}_e\mathbf{1}_e^\top$$

be the orthogonal projector onto $\mathbf{1}_e^\perp$, and define $D_c \coloneqq \Pi_\perp D$.

The centered direction D_c removes the shared-row component of D and captures the intrinsic variation across experts. This motivates router update maps built from the centered direction, for example through the centered left Gram matrix

$$D_c D_c^\top = \Pi_\perp D D^\top \Pi_\perp.$$

Definition G.1 (Router-compatible update maps)**.** A map $\mathcal{U}\colon \mathbb{R}^{e\times d} \to \mathbb{R}^{e\times d}$ is called *router-compatible* if it is expert-permutation equivariant, shared-row-shift invariant, and horizontal, namely,

$$\mathcal{U}(PD + \mathbf{1}_e a^\top) = P\,\mathcal{U}(D), \qquad \mathbf{1}_e^\top \mathcal{U}(D) = 0,$$

for all $D \in \mathbb{R}^{e\times d}$, all permutation matrices $P \in \mathbb{P}^e$, and all $a \in \mathbb{R}^d$.

The horizontality condition reflects the quotient geometry induced by softmax shift invariance. An update component in the shared-row direction $\mathbf{1}_e a^\top$ changes all expert logits by the same scalar and therefore does not affect the router probabilities. A geometry-compatible router update should therefore lie in the horizontal subspace

$$\{U \in \mathbb{R}^{e\times d} : \mathbf{1}_e^\top U = 0\}.$$

We consider two basic router-compatible update families. The first is a left-spectral update in the centered expert geometry:

$$\mathcal{U}_{\mathsf{L}}^{\mathsf{router}}(D) = \Psi(D_c D_c^\top) D_c, \qquad D_c = \Pi_\perp D,$$

where $\Psi\colon \mathbb{S}_+^e \to \mathbb{R}^{e\times e}$ is permutation equivariant and preserves the centered expert subspace. For example, the damped centered left-polar update corresponds to

$$\Psi(X) = (X + \varepsilon \Pi_\perp)^{-1/2}$$

on $\mathbf{1}_e^\perp$, or equivalently to applying the inverse square root on the nonzero centered expert subspace. Since left multiplication by a matrix acting on the centered expert space preserves $\mathbf{1}_e^\perp$, this update is horizontal:

$$\mathbf{1}_e^\top \mathcal{U}_{\mathsf{L}}^{\mathsf{router}}(D) = 0.$$

The second family is a centered row-norm update. Importantly, for row-norm maps it is not sufficient to center the input direction once. Although $D_c = \Pi_\perp D$ satisfies $\mathbf{1}_e^\top D_c = 0$, the rowwise-rescaled matrix

$$\mathrm{Diag}(\eta(\|D_{c,1:}\|_2), \dots, \eta(\|D_{c,e:}\|_2)) D_c$$

need not remain centered, because the scaling factors vary across expert rows. Therefore the router-compatible row-norm update is

$$\mathcal{U}_{\mathsf{row}}^{\mathsf{router}}(D) = \Pi_\perp\, \mathrm{Diag}(\eta(\|D_{c,1:}\|_2), \dots, \eta(\|D_{c,e:}\|_2)) D_c, \qquad D_c = \Pi_\perp D,$$

where $\eta\colon \mathbb{R}_+ \to \mathbb{R}$ is applied pointwise to the centered expert-row norms. The final projection is essential: it ensures that the actual parameter update, not only the input to the row-norm map, lies in the horizontal router subspace.

Proposition G.1 (Router-compatible update families)**.** *Assume that Ψ is permutation equivariant and maps the centered expert subspace to itself. Then the centered left-spectral update*

$$\mathcal{U}_{\mathsf{L}}^{\mathsf{router}}(D) = \Psi(D_c D_c^\top) D_c, \qquad D_c = \Pi_\perp D,$$

is router-compatible. Moreover, for any scalar function $\eta\colon \mathbb{R}_+ \to \mathbb{R}$, the projected centered row-norm update

$$\mathcal{U}_{\mathsf{row}}^{\mathsf{router}}(D) = \Pi_\perp\, \mathrm{Diag}(\eta(\|D_{c,1:}\|_2), \dots, \eta(\|D_{c,e:}\|_2)) D_c, \qquad D_c = \Pi_\perp D,$$

is router-compatible. In particular, for all $D \in \mathbb{R}^{e\times d}$, $P \in \mathbb{P}^e$, and $a \in \mathbb{R}^d$,

$$\mathcal{U}_{\mathsf{L}}^{\mathsf{router}}(PD + \mathbf{1}_e a^\top) = P\,\mathcal{U}_{\mathsf{L}}^{\mathsf{router}}(D), \qquad \mathcal{U}_{\mathsf{row}}^{\mathsf{router}}(PD + \mathbf{1}_e a^\top) = P\,\mathcal{U}_{\mathsf{row}}^{\mathsf{router}}(D),$$

and both updates satisfy

$$\mathbf{1}_e^\top \mathcal{U}_{\mathsf{L}}^{\mathsf{router}}(D) = 0, \qquad \mathbf{1}_e^\top \mathcal{U}_{\mathsf{row}}^{\mathsf{router}}(D) = 0.$$

Proof. We first record two elementary identities. Since $P \in \mathbb{P}^e$ satisfies $P\mathbf{1}_e = \mathbf{1}_e$ and $P^\top \mathbf{1}_e = \mathbf{1}_e$, the centering projector $\Pi_\perp = I_e - \frac{1}{e}\mathbf{1}_e\mathbf{1}_e^\top$ commutes with P, i.e., $\Pi_\perp P = P\Pi_\perp$. Moreover, for any $a \in \mathbb{R}^d$, $\Pi_\perp \mathbf{1}_e a^\top = 0$. Therefore, if $\widetilde{D} = PD + \mathbf{1}_e a^\top$, then its centered direction is

$$\widetilde{D}_c = \Pi_\perp \widetilde{D} = \Pi_\perp(PD + \mathbf{1}_e a^\top) = P\Pi_\perp D = PD_c.$$

We first consider the centered left-spectral update. Since $\widetilde{D}_c = PD_c$, we have $\widetilde{D}_c\widetilde{D}_c^\top = PD_cD_c^\top P^\top$. Using the permutation equivariance of Ψ,

$$\Psi(\widetilde{D}_c\widetilde{D}_c^\top) = \Psi(PD_cD_c^\top P^\top) = P\Psi(D_cD_c^\top)P^\top.$$

Hence

$$\mathcal{U}_{\mathsf{L}}^{\mathsf{router}}(\widetilde{D}) = \Psi(\widetilde{D}_c\widetilde{D}_c^\top)\widetilde{D}_c = P\Psi(D_cD_c^\top)P^\top PD_c = P\Psi(D_cD_c^\top)D_c = P\,\mathcal{U}_{\mathsf{L}}^{\mathsf{router}}(D).$$

This proves expert-permutation equivariance and shared-row-shift invariance for $\mathcal{U}_{\mathsf{L}}^{\mathsf{router}}$.

It remains to verify horizontality. Since $D_c = \Pi_\perp D$, every column of D_c lies in $\mathbf{1}_e^\perp$. By assumption, $\Psi(D_cD_c^\top)$ maps the centered expert subspace to itself. Therefore every column of $\Psi(D_cD_c^\top)D_c$ also lies in $\mathbf{1}_e^\perp$, and hence

$$\mathbf{1}_e^\top \mathcal{U}_{\mathsf{L}}^{\mathsf{router}}(D) = 0.$$

We next consider the projected centered row-norm update. Define

$$\mathsf{A}(D_c) \coloneqq \mathrm{Diag}(\eta(\|D_{c,1:}\|_2), \dots, \eta(\|D_{c,e:}\|_2)).$$

Since $\widetilde{D}_c = PD_c$, the row norms of $\widetilde{D}_c$ are the corresponding permutation of the row norms of D_c. Hence

$$\mathsf{A}(\widetilde{D}_c) = P\mathsf{A}(D_c)P^\top.$$

Using again that $\Pi_\perp P = P\Pi_\perp$, we obtain

$$\mathcal{U}_{\mathsf{row}}^{\mathsf{router}}(\widetilde{D}) = \Pi_\perp \mathsf{A}(\widetilde{D}_c)\widetilde{D}_c = \Pi_\perp P\mathsf{A}(D_c)P^\top PD_c = P\Pi_\perp \mathsf{A}(D_c)D_c = P\,\mathcal{U}_{\mathsf{row}}^{\mathsf{router}}(D).$$

Thus the projected centered row-norm update is also expert-permutation equivariant and shared-row-shift invariant.

Finally, horizontality of the row-norm update follows directly from the final projection:

$$\mathbf{1}_e^\top \mathcal{U}_{\mathsf{row}}^{\mathsf{router}}(D) = \mathbf{1}_e^\top \Pi_\perp \mathsf{A}(D_c)D_c = 0.$$

Therefore both update families are router-compatible. □

The converse is false in general: router compatibility does not force an update map to be left-spectral. Projected centered row-norm updates are already router-compatible, but they depend on individual centered expert-row norms rather than only on the centered Gram matrix $D_cD_c^\top$. Thus, left-spectral and row-norm updates should be viewed as two natural subclasses of router-compatible maps.

Hybrid router maps are obtained by composing router-compatible maps and projecting the final update back to the horizontal subspace. If $\mathcal{U}_1$ and $\mathcal{U}_2$ are router-compatible, then the horizontally projected composition

$$\Pi_\perp(\mathcal{U}_2 \circ \mathcal{U}_1)$$

is also router-compatible. Therefore, finite compositions of centered left-spectral and projected centered row-norm updates remain router-compatible. A representative row-after-left-spectral hybrid router update is

$$Z = \Psi(D_cD_c^\top)D_c, \qquad D_c = \Pi_\perp D,$$
$$\mathcal{U}_{\mathsf{hyb}}^{\mathsf{router}}(D) = \Pi_\perp\,\mathrm{Diag}(\eta(\|Z_{1:}\|_2), \dots, \eta(\|Z_{e:}\|_2))Z.$$

Such hybrid router optimizers combine the global expert-mixing geometry of left-spectral updates with the local expert-wise normalization of row-norm updates, while preserving expert-permutation equivariance, shared-row-shift invariance, and horizontality of the actual update.

Definition G.2 (Router-compatible optimizers)**.** A matrix optimizer for an MoE router is called *router-compatible* if its update rule has the form

$$(\forall k \in \mathbb{N}) \qquad W_{k+1} = W_k - \gamma_k \mathcal{U}(D_k),$$

where $\mathcal{U}$ is router-compatible and $D_k \in \mathbb{R}^{e\times d}$ is an update direction that transforms under expert permutations and shared row shifts in the same way as the gradient.

Table 1: **Optimizer classes across LLM layers induced by symmetry.** Each matrix parameter has a natural symmetry group, which determines the corresponding symmetry-compatible optimizer class. For LM heads and MoE routers, softmax shared-logit-shift invariance further induces quotient geometries; projected updates remove the shared-row direction so that the actual update acts on the non-redundant degrees of freedom.

Layer	Symmetry group	Optimizer classes
Linear / MLP / attention weights	$\mathbb{O}^{d_{\text{out}}} \times \mathbb{O}^{d_{\text{in}}}$	full spectral
Embedding	$\mathbb{P}^{v} \times \mathbb{O}^{d}$	LPRO: row-norm / right-spectral / hybrid
LM head	$\mathbb{P}^{v} \times \mathbb{O}^{d}$	LPRO: row-norm / right-spectral / hybrid
	$\mathbb{R}^{v\times d}/(\mathbf{1}_v\mathbb{R}^{1\times d})$	projected updates
SwiGLU MLP $(W_{\text{gate}}, W_{\text{up}}, W_{\text{down}}^{\top})$	$\mathbb{P}^{d_{\text{ff}}} \times \mathbb{O}^{d}$	LPRO: row-norm / right-spectral / hybrid
MoE router	$\mathbb{P}^{e} \times (\mathbf{1}_e\mathbb{R}^{1\times d})$	centered row-norm / left-spectral / hybrid
	$\mathbb{R}^{e\times d}/(\mathbf{1}_e\mathbb{R}^{1\times d})$	projected updates

Remark G.1 (Why the final projection matters). The final projection in the row-norm and hybrid router updates is not a numerical detail. Centering the direction $D_c = \Pi_{\perp} D$ removes the shared-row component before computing row norms, but row-dependent rescaling generally reintroduces a shared-row component. The final projection $\Pi_{\perp}$ removes this component and ensures that the update is horizontal. This is the router analogue of the horizontal update condition for untied LM heads under softmax shared-logit-shift invariance.

Remark G.2. Left- and right-spectral optimizers bear some formal resemblance to one-sided SHAMPOO [89, 148] and ASGO [6]. A key difference is that our framework applies the spectral transformation directly to the current update direction, or to a symmetry-compatible momentum direction, rather than maintaining moving averages of Gram matrices. More importantly, our design principle identifies which layer types such one-sided updates are appropriate for, according to the symmetry group of the corresponding parameter block.

Remark G.3 (Connection to non-Euclidean norm-based steepest descent). This symmetry-based construction differs from non-Euclidean steepest descent approaches based on choosing a matrix norm [12, 143, 152]. Row-wise or column-wise normalizations generally preserve only one-sided or permutation symmetries, rather than full bi-orthogonal symmetry. Thus, while such geometries may be useful in layer-specific settings, they should be distinguished from fully spectral constructions for ordinary matrix layers.

H A General and Unifying Symmetry-Induced Optimizer Geometry

Modern deep learning architectures contain many types of layers with matrix- or tensor-valued parameters, and these parameters may exhibit different symmetry structures. The left-permutation right-orthogonal symmetry of embeddings and LM heads, as well as the expert-permutation and shared-row-shift symmetry of MoE routers, are only two examples (see Table 1 for a summary). More generally, such symmetries arise when the layerwise loss is invariant under a group action on the parameter space. A natural optimizer should then respect the same symmetry through an equivariant update rule. In this subsection, we formulate this principle abstractly and recover the spectral, right-spectral, and left-spectral classes as canonical matrix instances.

Definition H.1 (Layer symmetry)**.** Let $W \in \mathbb{R}^{m\times n}$ denote a layer parameter and let $f\colon \mathbb{R}^{m\times n} \to \overline{\mathbb{R}}$ denote the layerwise loss function. Suppose a group $\mathcal{G}$ acts on the parameter space through transformations $W \mapsto g \cdot W$. We say that the parameterization admits symmetry group $\mathcal{G}$ if

$$(\forall g \in \mathcal{G}) \qquad f(g \cdot W) = f(W).$$

This symmetry induces an action on gradients.

Definition H.2 (Equivariant update rule)**.** Let $G = \nabla_W f(W)$. An update rule $\mathcal{U}(G)$ is said to be $\mathcal{G}$-equivariant if

$$(\forall g \in \mathcal{G}) \qquad \mathcal{U}(g \cdot G) = g \cdot \mathcal{U}(G).$$

Equivariant update rules avoid dependence on arbitrary coordinate systems that are invisible to the model. We recall the canonical matrix update classes induced by (bi-)orthogonal equivariance.

Theorem H.1 (Canonical matrix update classes induced by orthogonal equivariance)**.** *Let* $\mathcal{U}\colon \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ *be a continuous matrix-valued map.*

1. *If* $\mathcal{U}(PGQ^\top) = P\,\mathcal{U}(G)\,Q^\top$ *for all* $P \in \mathbb{O}^m$, $Q \in \mathbb{O}^n$, *then* $\mathcal{U}$ *is a spectral operator. Equivalently, if* $G = U_G \operatorname{Diag}(\sigma(G))V_G^\top$, *then* $\mathcal{U}(G) = U_G \operatorname{Diag}(\psi(\sigma(G)))V_G^\top$ *for some absolutely symmetric map* ψ.
2. *If* $\Psi\colon \mathbb{S}_+^n \to \mathbb{R}^{n\times n}$ *is an orthogonally equivariant spectral operator, then the map* $\mathcal{U}(G) = G\,\Psi(G^\top G)$ *is right-orthogonally equivariant, i.e.,* $\mathcal{U}(GQ^\top) = \mathcal{U}(G)Q^\top$ *for all* $Q \in \mathbb{O}^n$.
3. *If* $\Phi\colon \mathbb{S}_+^m \to \mathbb{R}^{m\times m}$ *is an orthogonally equivariant spectral operator, then the map* $\mathcal{U}(G) = \Phi(GG^\top)G$ *is left-orthogonally equivariant, i.e.,* $\mathcal{U}(PG) = P\,\mathcal{U}(G)$ *for all* $P \in \mathbb{O}^m$.

Proof.

1. This is precisely Theorem 3.2. Indeed, bi-orthogonal equivariance implies that $\mathcal{U}$ is a spectral operator, so for any singular value decomposition $G = U_G \operatorname{Diag}(\sigma(G))V_G^\top$, there exists an absolutely symmetric map ψ such that $\mathcal{U}(G) = U_G \operatorname{Diag}(\psi(\sigma(G)))V_G^\top$.

2. Let $\mathcal{U}(G) = G\,\Psi(G^\top G)$, where $\Psi\colon \mathbb{S}_+^n \to \mathbb{R}^{n\times n}$ satisfies $\Psi(QXQ^\top) = Q\,\Psi(X)Q^\top$ for all $X \in \mathbb{S}_+^n$ and $Q \in \mathbb{O}^n$. Then for any $Q \in \mathbb{O}^n$, $(GQ^\top)^\top(GQ^\top) = QG^\top GQ^\top$. Hence, by equivariance of Ψ,

$$\Psi((GQ^\top)^\top(GQ^\top)) = \Psi(QG^\top GQ^\top) = Q\,\Psi(G^\top G)\,Q^\top.$$

   Therefore,

$$\begin{aligned}\mathcal{U}(GQ^\top) &= GQ^\top\,\Psi((GQ^\top)^\top(GQ^\top)) \\ &= GQ^\top\big(Q\,\Psi(G^\top G)\,Q^\top\big) = G\,\Psi(G^\top G)\,Q^\top = \mathcal{U}(G)Q^\top.\end{aligned}$$

   Thus $\mathcal{U}$ is right-orthogonally equivariant.

3. Let $\mathcal{U}(G) = \Phi(GG^\top)\,G$, where $\Phi\colon \mathbb{S}_+^m \to \mathbb{R}^{m\times m}$ satisfies $\Phi(PXP^\top) = P\,\Phi(X)P^\top$ for all $X \in \mathbb{S}_+^m$ and $P \in \mathbb{O}^m$. Then for any $P \in \mathbb{O}^m$, $(PG)(PG)^\top = PGG^\top P^\top$. Hence, by equivariance of Φ,

$$\Phi((PG)(PG)^\top) = \Phi(PGG^\top P^\top) = P\,\Phi(GG^\top)\,P^\top.$$

   Therefore,

$$\mathcal{U}(PG) = \Phi((PG)(PG)^\top)\,PG = \big(P\,\Phi(GG^\top)\,P^\top\big)PG = P\,\Phi(GG^\top)\,G = P\,\mathcal{U}(G).$$

   Thus $\mathcal{U}$ is left-orthogonally equivariant.

□

Remark H.1. Unlike the bi-orthogonal case, right-orthogonal or left-orthogonal equivariance alone does not characterize the right-spectral or left-spectral forms. For example, any map of the form $\mathcal{U}(G) = \mathsf{A}(GG^\top)G$ with arbitrary continuous $\mathsf{A}\colon \mathbb{S}_+^m \to \mathbb{R}^{m\times m}$ is right-orthogonally equivariant, but need not be right-spectral. Similarly, any map of the form $\mathcal{U}(G) = G\,\mathsf{B}(G^\top G)$ with arbitrary continuous $\mathsf{B}\colon \mathbb{S}_+^n \to \mathbb{R}^{n\times n}$ is left-orthogonally equivariant, but need not be left-spectral. Thus, the right-spectral and left-spectral classes should be viewed as canonical symmetry-compatible subclasses rather than exhaustive characterizations.

The canonical normalized representatives of these classes are obtained by choosing the inverse-square-root operator on the relevant Gram matrix, with Moore–Penrose pseudoinverse when necessary. The corresponding normalized updates are $\Delta W = U_G V_G^\top$ for full spectral, $\Delta W = G(G^\top G)^{\dagger/2}$ for right-spectral, and $\Delta W = (GG^\top)^{\dagger/2}G$ for left-spectral. These canonical updates normalize gradient anisotropy along the symmetry axes while preserving the invariant geometry of the model.

H.1 Symmetry-to-Optimizer Principle and Architecture–Optimizer Co-Design

The preceding developments suggest a unifying principle for optimizer design in modern deep learning architectures: the optimizer geometry should be determined by the symmetry structure of the underlying parameterization. This does not require the full layerwise loss to be globally invariant under every group action considered. Rather, it is enough that the gradient, or more generally the chosen update direction, transforms equivariantly under the relevant representation of the parameter block. The optimizer update should then transform in the same way, and should lie in the corresponding quotient or horizontal subspace whenever the parameterization contains symmetry-redundant directions.

Accordingly, whenever the update direction associated with a layer parameter W transforms under a symmetry group $\mathcal{G}$, a natural optimizer should use an update map $\mathcal{U}$ that is equivariant under the same induced action. For matrix-valued parameters, this principle leads to several canonical optimizer geometries: full spectral updates under bi-orthogonal symmetry; right-spectral updates under right-orthogonal symmetry; left-spectral updates under left-orthogonal symmetry; row-norm and hybrid updates under left-permutation/right-orthogonal symmetry; and projected centered left-spectral or row-norm updates under the expert-permutation and shared-row-shift quotient geometry of MoE routers. For LM heads, the same softmax shared-logit-shift invariance induces a horizontal quotient constraint on the vocabulary rows. Thus, the symmetry structure of the layer determines not only the appropriate optimizer class, but also whether the final update should be projected to remove redundant directions.

This principle also gives a practical recipe for architecture–optimizer co-design. Given a new architecture block, one should:

1. identify the symmetry group of the parameterization;
2. determine whether the symmetry acts on the left, on the right, on both sides, or only after quotienting out symmetry-redundant directions;
3. choose the matching symmetry-compatible optimizer class;
4. project the update onto the horizontal subspace when the parameterization has redundant shift directions; and
5. use the smallest Gram matrix or invariant statistic compatible with that symmetry.

In Section J.1, we instantiate this recipe by extending momentum PolarGrad [83] to one-sided and hybrid variants, including LeftPolarGradM, RightPolarGradM, RowNormM, and HybridPolarGradM.

Remark H.2 (Projected and proximal extensions). The symmetry-compatible optimizer classes above extend naturally to projected and proximal updates. If $\mathcal{T}(D_k)$ is $\mathcal{G}$-equivariant and the constraint set $\mathcal{C}$ is $\mathcal{G}$-invariant, then the projected update

$$W_{k+1} = \mathrm{proj}_{\mathcal{C}}\big(W_k - \gamma_k \mathcal{T}(D_k)\big)$$

is symmetry-compatible, since the Euclidean projection onto a $\mathcal{G}$-invariant set is $\mathcal{G}$-equivariant whenever it is uniquely defined. Similarly, if h is a $\mathcal{G}$-invariant regularizer, then

$$W_{k+1} = \mathrm{prox}_{\gamma_k h}\big(W_k - \gamma_k \mathcal{T}(D_k)\big)$$

preserves the same symmetry whenever the proximal map is uniquely defined. In the bi-orthogonal setting, this includes unitarily invariant constraints and regularizers, such as spectral-norm, nuclear-norm, rank, and Schatten-p constraints or penalties. We leave a systematic study of such variants, including ProxPolarGrad, to future work.

I Proofs of Main Text

Proof of (1). Let $G = U\Sigma V^\top$ be a singular value decomposition of G. Then

$$PGQ^\top = (PU)\Sigma(QV)^\top.$$

Since P and Q are orthogonal, PU and QV are again orthogonal matrices. Hence $(PU)\Sigma(QV)^\top$ is a singular value decomposition of $PGQ^\top$. Therefore, we have

$$\mathrm{polar}(PGQ^\top) = (PU)(QV)^\top = PUV^\top Q^\top = P\,\mathrm{polar}(G)Q^\top.$$

□

Proof of Proposition 2.1. We also state the desired results for Polyak and Nesterov momentum.

For EMA momentum, we define $M_k = \beta M_{k-1} + (1-\beta)G_k$ with $M_{-1} = 0$, and, for the transformed sequence, $\widetilde{M}_k = \beta\widetilde{M}_{k-1} + (1-\beta)\widetilde{G}_k$ with $\widetilde{M}_{-1} = 0$. Then we have $\widetilde{M}_k = PM_kQ^\top$ and $\operatorname{polar}(\widetilde{M}_k) = P\operatorname{polar}(M_k)Q^\top$.

For Polyak heavy-ball momentum, we define $M_k = \beta M_{k-1} + G_k$ with $M_{-1} = 0$, and $\widetilde{M}_k = \beta\widetilde{M}_{k-1} + \widetilde{G}_k$ with $\widetilde{M}_{-1} = 0$. Then we have $\widetilde{M}_k = PM_kQ^\top$ and $\operatorname{polar}(\widetilde{M}_k) = P\operatorname{polar}(M_k)Q^\top$.

We first prove the stated transformation for the momentum sequences. For the EMA recursion, assume inductively that $\widetilde{M}_{k-1} = PM_{k-1}Q^\top$. Using the bi-orthogonal equivariance of the gradient, we have

$$\widetilde{G}_k = \nabla_W f(\widetilde{W}_k) = \nabla_W f(PW_kQ^\top) = P\nabla_W f(W_k)Q^\top = PG_kQ^\top.$$

Hence

$$\begin{aligned}\widetilde{M}_k = \beta\widetilde{M}_{k-1} + (1-\beta)\widetilde{G}_k &= \beta PM_{k-1}Q^\top + (1-\beta)PG_kQ^\top \\ &= P(\beta M_{k-1} + (1-\beta)G_k)Q^\top = PM_kQ^\top.\end{aligned}$$

Thus the EMA momentum is bi-orthogonally equivariant. The Polyak case is identical, replacing $(1-\beta)$ by 1, which gives $\widetilde{M}_k = PM_kQ^\top$.

For Nesterov momentum, the momentum recursion is the same as in the Polyak case, so $\widetilde{M}_k = PM_kQ^\top$. Therefore,

$$\widetilde{N}_k = \widetilde{G}_k + \beta\widetilde{M}_k = PG_kQ^\top + \beta PM_kQ^\top = P(G_k + \beta M_k)Q^\top = PN_kQ^\top.$$

All claims about orthogonal polar factors follow from (1). For example,

$$\operatorname{polar}(\widetilde{N}_k) = \operatorname{polar}(PN_kQ^\top) = P\operatorname{polar}(N_k)Q^\top.$$

The proofs for the momentum polar factors are identical. □

Proof of Theorem 3.2. Suppose first that $\mathfrak{U}$ is of the stated spectral form. Let $P \in \mathbb{O}^m$ and $Q \in \mathbb{O}^n$ be orthogonal. If $D = U\operatorname{Diag}(\sigma(D))V^\top$, then $PDQ^\top = (PU)\operatorname{Diag}(\sigma(D))(QV)^\top$ is a singular value decomposition of $PDQ^\top$. Therefore, we have

$$\mathfrak{U}(PDQ^\top) = (PU)\operatorname{Diag}(\psi(\sigma(D)))(QV)^\top = P\,\mathfrak{U}(D)Q^\top,$$

so $\mathfrak{U}$ is bi-orthogonally equivariant.

Conversely, suppose that $\mathfrak{U}$ is bi-orthogonally equivariant. Let $D = U\operatorname{Diag}(\sigma(D))V^\top$ be a singular value decomposition of D. Then we have

$$\mathfrak{U}(D) = \mathfrak{U}(U\operatorname{Diag}(\sigma(D))V^\top) = U\mathfrak{U}(\operatorname{Diag}(\sigma(D)))V^\top.$$

Thus the action of $\mathfrak{U}$ is completely determined by its action on diagonal matrices of singular values. Since bi-orthogonal equivariance ensures $\mathfrak{U}$ commutes with arbitrary diagonal sign matrices, forcing all off-diagonal elements to be zero, let us define $\operatorname{Diag}(\psi(s)) \coloneqq \mathfrak{U}(\operatorname{Diag}(s))$, where $s \in \mathbb{R}^r_+$. The bi-orthogonal equivariance of $\mathfrak{U}$ further implies that this definition is independent of the particular singular value decomposition and that ψ is absolutely symmetric with respect to permutations and sign changes compatible with singular-value representations. Hence we have $\mathfrak{U}(D) = U\operatorname{Diag}(\psi(\sigma(D)))V^\top$, which is exactly the claimed spectral form. □

Proof of Theorem 4.1. Let $P \in \mathbb{P}^v$, $R \in \mathbb{O}^d$, and $D \in \mathbb{R}^{v\times d}$. Since P is a permutation matrix, we have $P^\top P = I_v$. Therefore,

$$(PDR^\top)^\top(PDR^\top) = RD^\top P^\top PDR^\top = RD^\top DR^\top.$$

Using the orthogonal equivariance of Φ, it follows that

$$\Phi((PDR^\top)^\top(PDR^\top)) = \Phi(RD^\top DR^\top) = R\Phi(D^\top D)R^\top.$$

Hence we obtain

$$\begin{aligned}\mathfrak{U}_{\mathsf{R}}(PDR^\top) = PDR^\top\Phi((PDR^\top)^\top(PDR^\top)) &= PDR^\top(R\,\Phi(D^\top D)R^\top) \\ &= PD\Phi(D^\top D)R^\top = P\,\mathfrak{U}_{\mathsf{R}}(D)R^\top.\end{aligned}$$

□

Proof of Proposition 4.2. For any $D \in \mathbb{R}^{v\times d}$, permutation matrix $P \in \mathbb{P}^v$, and orthogonal matrix $R \in \mathbb{O}^d$,

$$(\mathcal{U}_2 \circ \mathcal{U}_1)(PDR^\top) = \mathcal{U}_2(\mathcal{U}_1(PDR^\top)) = \mathcal{U}_2(P\,\mathcal{U}_1(D)R^\top) = P\,\mathcal{U}_2(\mathcal{U}_1(D))R^\top.$$

Hence $\mathcal{U}_2 \circ \mathcal{U}_1$ is left-permutation and right-orthogonal equivariant. □

J Implementation Details of Practical Optimizers

We provide the implementation details of our proposed practical optimizers in this section.

J.1 Practical Optimizers for Embeddings, LM Heads, and MoE Routers

Section 4 introduced three main classes of symmetry-compatible optimizers: one-sided spectral optimizers, row-norm-based optimizers, and hybrid variants obtained by composing row-wise normalization with one-sided spectral updates. We now describe their practical momentum implementations. The main computational issue is the efficient and stable approximation of matrix inverse square roots, or equivalently orthogonal polar factors, which we compute using GPU-friendly numerical linear algebra routines. In addition, for untied LM heads and MoE routers, the practical updates must respect the horizontal quotient geometry induced by softmax shared-logit-shift invariance. Thus, when row-wise nonlinear operations are used for these blocks, we project the final update back to the corresponding centered subspace.

J.1.1 One-Sided Spectral Optimizers

For embedding, LM head, and SwiGLU MLP projection matrices, the relevant matrices are often tall-skinny. In this regime, right-spectral updates are attractive because they only require the inverse square root of the smaller right Gram matrix

$$C_k \coloneqq G_k^\top G_k \qquad \text{or, with momentum,} \qquad C_k \coloneqq M_k^\top M_k.$$

For an untied LM head, we use the centered momentum direction

$$\Pi_v^\perp \coloneqq I_v - \frac{1}{v}\mathbf{1}_v\mathbf{1}_v^\top, \qquad M_{k,c} \coloneqq \Pi_v^\perp M_k,$$

so that the update removes the shared vocabulary-logit-shift direction. The corresponding right Gram matrix is

$$C_k \coloneqq M_{k,c}^\top M_{k,c}.$$

For MoE routers, the corresponding left-spectral update is applied in the centered expert geometry. Writing

$$\Pi_\perp \coloneqq I_e - \frac{1}{e}\mathbf{1}_e\mathbf{1}_e^\top, \qquad M_{k,c} \coloneqq \Pi_\perp M_k,$$

the relevant Gram matrix is

$$C_k \coloneqq M_{k,c}M_{k,c}^\top.$$

To compute the inverse square roots in these updates, we use Newton–Schulz iterations with the polynomial coefficients of Polar Express [5]. This is motivated by the connection between polynomial iterations for polar decomposition and inverse-square-root computation [62]. For numerical stability, the Gram inverse-square-root implementation is performed in `float32`. Other fast inverse-square-root routines, such as PRISM [157], could be used in the same role. For RightPolarGradM, we also provide a Gram Newton–Schulz implementation [164], which directly approximates

$$M_k(M_k^\top M_k)^{-1/2}$$

and supports more inner iterations in lower-precision formats such as `bfloat16` or `float16`. The resulting one-sided momentum polar-gradient algorithms are summarized in Algorithm J.1.

At the level of exact polar decomposition, LeftPolarGradM and RightPolarGradM compute the same polar direction whenever both sides are well-defined. Their distinction is therefore computational and architectural: they differ in which Gram matrix is formed, which inverse square root is computed, and which layer symmetry they are intended to respect. For LM heads and MoE routers, the one-sided spectral updates are applied to centered momentum directions when the corresponding softmax shared-logit-shift quotient geometry is present.

Algorithm J.1 LEFTPOLARGRADM and RIGHTPOLARGRADM

Input: $W_0 \in \mathbb{R}^{m\times n}$, $M_{-1} = 0$, learning rates $\{\gamma_k\}_{k\geqslant 0}$, momentum $\beta \in [0, 1)$, scaling exponent $\alpha \in [0, 1]$, damping $\varepsilon > 0$, weight decay $\lambda \geqslant 0$

for $k = 0, \ldots, K-1$ **do**
  $G_k = \nabla_W f(W_k)$
  $M_k = \beta M_{k-1} + (1-\beta)G_k$
  if LEFTPOLARGRADM **then**
    $C_k = M_k M_k^\top$
    $L_k = (C_k + \varepsilon I)^{-1/2}$ via POLAR EXPRESS or Newton–Schulz iteration
    $\nu_k = \max\{\mathrm{tr}(C_k L_k), \varepsilon\}$
    $W_{k+1} = (1-\gamma_k\lambda)W_k - \gamma_k \nu_k^\alpha L_k M_k$
  else if RIGHTPOLARGRADM **then**
    $C_k = M_k^\top M_k$
    $R_k = (C_k + \varepsilon I)^{-1/2}$ via POLAR EXPRESS or Newton–Schulz iteration
    $\nu_k = \max\{\mathrm{tr}(C_k R_k), \varepsilon\}$
    $W_{k+1} = (1-\gamma_k\lambda)W_k - \gamma_k \nu_k^\alpha M_k R_k$
  end if
end for

Return: W_K

J.1.2 Row-Norm-Based and Hybrid Variants

For embedding, LM head, and SwiGLU MLP projection matrices, row-norm-based updates provide a cheaper LPRO-compatible alternative. Given a momentum direction M_k, define

$$D_\eta(M_k) \coloneqq \mathrm{Diag}(\eta(\|M_{k,1:}\|_2), \ldots, \eta(\|M_{k,m:}\|_2)).$$

For embeddings and SwiGLU gate/up projections, a row-norm update takes the form

$$\mathcal{T}_{\mathsf{row}}(M_k) = D_\eta(M_k) M_k.$$

Here η may be chosen as a bounded row-scaling rule or as a smoothed normalization rule such as $\eta(t) = 1/(t+\varepsilon)$.

For untied LM heads, the update must also remove the shared vocabulary-logit-shift direction. Let

$$\Pi_v^\perp \coloneqq I_v - \frac{1}{v}\mathbf{1}_v\mathbf{1}_v^\top, \qquad M_{k,c} \coloneqq \Pi_v^\perp M_k.$$

The horizontal LM head row-norm update is

$$\mathcal{T}_{\mathsf{row}}^{\mathrm{LM}}(M_k) = \Pi_v^\perp D_\eta(M_{k,c}) M_{k,c}.$$

The final projection is essential for row-norm updates: even though $\mathbf{1}_v^\top M_{k,c} = 0$, row-dependent rescaling by $D_\eta(M_{k,c})$ need not preserve the zero-row-mean constraint.

Hybrid variants combine row-wise scaling with a one-sided spectral step. For embeddings and SwiGLU gate/up projections, there are two natural orders:

$$\text{right-spectral/row-norm:} \qquad Z_k = M_k(M_k^\top M_k + \varepsilon I)^{-1/2}, \qquad \mathcal{T}_{\mathsf{hyb}}(M_k) = D_\eta(Z_k) Z_k,$$

and

$$\text{row-norm/right-spectral:} \qquad Z_k = D_\eta(M_k) M_k, \qquad \mathcal{T}_{\mathsf{hyb}}(M_k) = Z_k(Z_k^\top Z_k + \varepsilon I)^{-1/2}.$$

Both remain left-permutation and right-orthogonal equivariant. For LM heads, the same constructions are applied to the centered momentum $M_{k,c} = \Pi_v^\perp M_k$, with horizontal projections inserted after rowwise nonlinear stages and on the final update. For example, the row-norm/right-spectral order becomes

$$Z_k = \Pi_v^\perp D_\eta(M_{k,c}) M_{k,c}, \qquad \mathcal{T}_{\mathsf{hyb}}^{\mathrm{LM}}(M_k) = \Pi_v^\perp Z_k(Z_k^\top Z_k + \varepsilon I)^{-1/2}.$$

For MOE routers, the same principle applies in the centered expert geometry. Writing

$$\Pi_\perp \coloneqq I_e - \frac{1}{e}\mathbf{1}_e\mathbf{1}_e^\top, \qquad M_{k,c} = \Pi_\perp M_k,$$

the router row-norm update is

$$\mathcal{T}_{\text{row}}^{\text{router}}(M_k) = \Pi_{\perp} D_\eta(M_{k,c}) M_{k,c}.$$

Hybrid router updates combine a centered left-spectral step with projected row-wise normalization across experts. We refer to these practical hybrid variants collectively as HYBRIDPOLARGRADM. The algorithms for ROWNORMM and HYBRIDPOLARGRADM are summarized in Algorithm J.2.

Algorithm J.2 ROWNORMM and HYBRIDPOLARGRADM with Optional Horizontal Projection

Input: W_0, $M_{-1} = 0$, learning rates $\{\gamma_k\}_{k \geqslant 0}$, momentum $\beta \in [0, 1)$, scaling exponent $\alpha \in [0, 1]$, row-scaling rule η, damping $\varepsilon > 0$, weight decay $\lambda \geqslant 0$

for $k = 0, \ldots, K-1$ **do**
  $G_k = \nabla_W f(W_k)$
  $M_k = \beta M_{k-1} + (1-\beta) G_k$
  Set the working direction $\overline{M}_k$ and final projector Π by layer type:

$$(\overline{M}_k, \Pi) = \begin{cases} (M_k, I), & \text{embedding or SwiGLU gate/up,} \\ (\Pi_v^{\perp} M_k, \Pi_v^{\perp}), & \text{LM head,} \\ (\Pi_e^{\perp} M_k, \Pi_e^{\perp}), & \text{MOE router,} \end{cases}$$

  where

$$\Pi_v^{\perp} = I_v - \frac{1}{v} \mathbf{1}_v \mathbf{1}_v^{\top}, \qquad \Pi_e^{\perp} = I_e - \frac{1}{e} \mathbf{1}_e \mathbf{1}_e^{\top}.$$

  if ROWNORMM **then**
    $D_k = \mathrm{Diag}(\eta(\|\overline{M}_{k,1:}\|_2), \ldots, \eta(\|\overline{M}_{k,m:}\|_2))$
    $U_k = \Pi D_k \overline{M}_k$
  else if HYBRIDPOLARGRADM, right-spectral/row-norm order **then**
    $C_k = \overline{M}_k^{\top} \overline{M}_k$
    $R_k = (C_k + \varepsilon I)^{-1/2}$ via POLAR EXPRESS or Newton–Schulz iteration
    $\nu_k = \max\{\mathrm{tr}(C_k R_k), \varepsilon\}$
    $Z_k = \Pi \nu_k^{\alpha} \overline{M}_k R_k$
    $D_k = \mathrm{Diag}(\eta(\|Z_{k,1:}\|_2), \ldots, \eta(\|Z_{k,m:}\|_2))$
    $U_k = \Pi D_k Z_k$
  else if HYBRIDPOLARGRADM, row-norm/right-spectral order **then**
    $D_k = \mathrm{Diag}(\eta(\|\overline{M}_{k,1:}\|_2), \ldots, \eta(\|\overline{M}_{k,m:}\|_2))$
    $Z_k = \Pi D_k \overline{M}_k$
    $C_k = Z_k^{\top} Z_k$
    $R_k = (C_k + \varepsilon I)^{-1/2}$ via POLAR EXPRESS or Newton–Schulz iteration
    $\nu_k = \max\{\mathrm{tr}(C_k R_k), \varepsilon\}$
    $U_k = \Pi \nu_k^{\alpha} Z_k R_k$
  else if MOE router and HYBRIDPOLARGRADM, left-spectral/row-norm order **then**
    $C_k = \overline{M}_k \overline{M}_k^{\top}$
    $L_k = (C_k + \varepsilon I)^{-1/2}$ via POLAR EXPRESS or Newton–Schulz iteration
    $\nu_k = \max\{\mathrm{tr}(C_k L_k), \varepsilon\}$
    $Z_k = \Pi \nu_k^{\alpha} L_k \overline{M}_k$
    $D_k = \mathrm{Diag}(\eta(\|Z_{k,1:}\|_2), \ldots, \eta(\|Z_{k,e:}\|_2))$
    $U_k = \Pi D_k Z_k$
  end if
  $W_{k+1} = (1 - \gamma_k \lambda) W_k - \gamma_k U_k$
end for
Return: W_K

For down projections in SwiGLU MLPs, we apply the same procedure to $W_{\text{down}}^{\top}$ and transpose the resulting update back. For routers, the hybrid update uses the left-spectral branch; for embeddings, LM heads, and SwiGLU gate/up projections, it uses the right-spectral branches.

The practical distinction among these variants is both geometric and computational. RIGHTPOLARGRADM preserves the right-orthogonal geometry through a Gram inverse square root, while

ROWNORMM is a purely row-adaptive LPRO-compatible alternative. HYBRIDPOLARGRADM interpolates between the two by composing row normalization and one-sided spectral normalization. For LM heads and MOE routers, softmax shared-logit-shift invariance imposes an additional quotient geometry; therefore, row-norm and hybrid updates are projected back to the horizontal subspace so that the actual update removes the shared-row direction. For MOE routers, the same alternatives appear in the centered expert geometry: one may use a projected centered row-norm update, a centered left-spectral update, or a projected hybrid left-spectral/row-norm update.

J.2 Numerical Algorithm for Matrix Inverse Square Root via POLAR EXPRESS

We illustrate below the algorithm for computing matrix inverse square root via POLAR EXPRESS, which is based on the intrinsic connection between polynomial iterations for computing the orthogonal polar factor and those for computing the inverse square root of a square matrix, stated in the following theorem.

Theorem J.1 (Higham [62]). *Let $A \in \mathbb{R}^{n\times n}$ be a square matrix with nonnegative eigenvalues. Consider any iteration of the form $X_{k+1} = X_k h(X_k^2)$ that converges to $\mathrm{msgn}(X_0)$ for $X_0 = \left(\begin{smallmatrix} 0 & A \\ I_n & 0 \end{smallmatrix}\right)$ with order of convergence q. Then in the coupled iteration $X_{k+1} = X_k h(Y_k X_k)$, $Y_{k+1} = h(Y_k X_k) Y_k$, with $X_0 = A$ and $Y_0 = I_n$, we have $X_k \to A^{1/2}$ and $Y_k \to A^{-1/2}$, both with order of convergence q. Here* msgn *denotes the matrix sign function.*

Algorithm J.3 Matrix Inverse Square Root via POLAR EXPRESS [5]

Input: $A \in \mathbb{R}^{n\times n}$, $\varepsilon > 0$, $K \in \mathbb{N}^*$, sequence of POLAR EXPRESS coefficients $\{(a_k, b_k, c_k)\}_{k=1}^K$

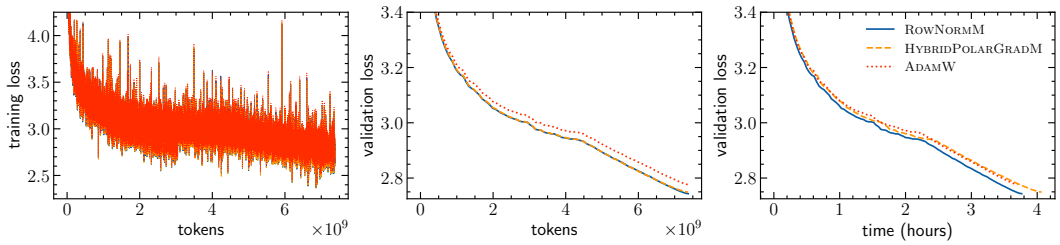
$A = (A + A^\top)/2 + \varepsilon I_n$
$\alpha = 1.02\|\|A\|\|_{\mathrm{F}} + \varepsilon$
$Y_1 = A/\alpha$
$Z_1 = I_n$
for $k = 1, \ldots, K$ **do**
  $(\overline{a}_k, \overline{b}_k, \overline{c}_k) = (a_k + b_k + c_k, -b_k - 2c_k, c_k)$
  $R_k = I_n - Z_k Y_k$
  $R_k = (R_k + R_k^\top)/2$ (optional matrix symmetrization)
  $M_k = \overline{a}_k I_n + \overline{b}_k R_k + \overline{c}_k R_k^2$
  $Y_{k+1} = Y_k M_k$
  $Z_{k+1} = M_k Z_k$
end for
Return: $Z_{K+1}/\sqrt{\alpha}$

K Additional Numerical Experiments

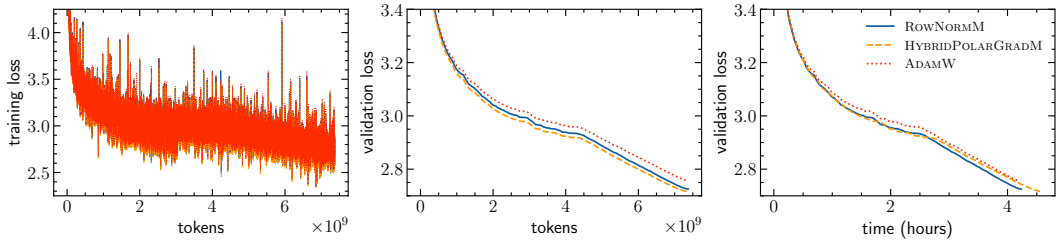
We provide experimental results for Gemma 3 1B-style, Qwen3-0.6B-style and OLMOE-1B-7B-style pre-training.

K.1 Gemma 3 1B-Style Pre-Training

We pre-train a Gemma 3 1B-style dense language model [47]. Compared with the Qwen3-0.6B-style experiment, this model has both a larger vocabulary size of 262,144, and a larger hidden dimension of 1152. Consequently, the input embedding and LM head matrices are substantially larger, and their matrix gradients may be more anisotropic or ill-conditioned. This setting therefore provides a more stringent test of geometry-aware optimizers for vocabulary-indexed matrix parameters. For right-spectral updates, such as those used inside RIGHTPOLARGRADM and HYBRIDPOLARGRADM, the larger hidden dimension also makes the Gram matrix inverse-square-root computation more demanding, often requiring more accurate or additional Newton–Schulz iterations. To keep the total model size close to one billion trainable parameters after using untied embeddings, we reduce the number of hidden layers from 26 to 18, resulting in 1,087,138,944 trainable parameters. We use the same three optimizer assignments for the input embedding and LM head matrices as in the Qwen3-0.6B-style experiment, and compare two assignments for the SwiGLU MLP projection matrices: (a) MUON and (b) HYBRIDPOLARGRADM.



(a) SwiGLU MLP projection matrices use MUON, equivalently RIGHTPOLARGRADM with $\alpha = 0$.



(b) SwiGLU MLP projection matrices use HYBRIDPOLARGRADM with a row-norm/right-spectral composition.

Figure 4: Training and validation losses for Gemma 3 1B-style pre-training. In each subfigure, the three configurations differ only in the optimizer assigned to the input embedding and LM head matrices: ROWNORMM, HYBRIDPOLARGRADM, or ADAMW.

For configurations (i)–(iii), the final validation losses are 2.7434, 2.7492, and 2.7761 in Figure 4a, compared with 2.7251, 2.7160, and 2.7562 in Figure 4b, respectively. The best embedding/LM head assignment is ROWNORMM when the MLP projections use MUON, and HYBRIDPOLARGRADM when they use hybrid updates. Both assignments yield substantially lower final validation losses than ADAMW under either MLP optimizer. These findings are consistent with the hypothesis that geometry-compatible updates are particularly beneficial for large vocabulary-indexed matrices whose gradients may be anisotropic or poorly conditioned.

The comparison between Figure 4a and Figure 4b also shows that replacing MUON with HYBRIDPOLARGRADM for the SwiGLU MLP projections benefits all three embedding/LM head configurations. The final validation losses fall from 2.7434 to 2.7251 with ROWNORMM, from 2.7492 to 2.7160 with HYBRIDPOLARGRADM, and from 2.7761 to 2.7562 with ADAMW. A possible explanation lies in the pronounced aspect ratio of the gate and up projection matrices. In Gemma 3 1B-style models, $d_{\text{model}} = 1152 \ll d_{\text{ff}} = 6912$, giving these matrices six times as many rows as columns, with each row associated with an intermediate neuron. In addition to the right-spectral transformation used by MUON, HYBRIDPOLARGRADM applies row normalization before the spectral step, potentially mitigating the effects of unequal gradient scales across intermediate neurons.

At the same time, HYBRIDPOLARGRADM incurs higher computational overhead than ROWNORMM because it requires approximating matrix inverse square roots through inner Gram Newton–Schulz iterations. In contrast, ROWNORMM performs only row-wise normalization and therefore avoids spectral computations altogether. Thus, the Gemma 3 1B-style experiment highlights a practical tradeoff: the hybrid row-norm/right-spectral update achieves the best validation loss, while the row-norm-only update provides a computationally cheaper symmetry-compatible alternative that still improves markedly over the coordinate-wise ADAMW baseline.

K.1.1 Gemma 3 1B-Style Pre-Training Learning Rate Sweep

For this pre-training experiment, we also perform a base learning rate sweep for the embedding and LM head matrices, keeping the learning rates for scalars/vector and matrices fixed. For simplicity, we keep both base learning rates for the embedding and LM head matrices to be equal, although more delicate tuning is possible.

We observe from Table 2 that the validation loss gaps between configuration (iii) and configurations (i)–(ii) remain substantial across the swept base learning rates. As shown in Figure 5, the separation

Table 2: Base learning-rate sweep for the input embedding and LM head matrices in Gemma 3 1B-style pre-training. We sweep only $\gamma_{0,\mathrm{emb}} = \gamma_{0,\mathrm{head}}$. In setting (a), SwiGLU MLP projection matrices use MUON; in setting (b), they use HYBRIDPOLARGRADM with a row-norm/right-spectral composition.

Setting	Embedding/LM head optimizer	$\gamma_{0,\mathrm{emb}} = \gamma_{0,\mathrm{head}}$	Final validation loss	Training time (hr)
(a) SwiGLU MLP: MUON	ROWNORMM	0.001	2.7567	3.721
		0.0025	2.7483	3.708
		0.005	2.7453	3.709
		0.01	**2.7434**	3.749
	HYBRIDPOLARGRADM (row-norm/right-spectral, $\alpha = 1$)	0.00005	**2.7492**	4.050
		0.0001	2.7496	4.080
		0.00025	2.7531	4.128
		0.0005	2.7713	4.086
	ADAMW	0.00025	2.7855	3.690
		0.0005	2.7781	3.692
		0.001	**2.7761**	3.682
		0.0025	2.7776	3.654
(b) SwiGLU MLP: HYBRIDPOLARGRADM	ROWNORMM	0.001	2.7467	4.220
		0.0025	2.7385	4.164
		0.005	**2.7251**	4.216
		0.01	2.7251	4.267
	HYBRIDPOLARGRADM (row-norm/right-spectral, $\alpha = 1$)	0.00005	**2.7160**	4.566
		0.0001	2.7199	4.616
		0.00025	2.7252	4.620
		0.0005	2.7362	4.589
	ADAMW	0.00025	2.7790	4.152
		0.0005	2.7654	4.164
		0.001	**2.7562**	4.191
		0.0025	2.7603	4.029

between the ADAMW embedding/LM head baselines and the symmetry-compatible alternatives is not explained by a single learning-rate choice. Across the swept values of $\gamma_{0,\mathrm{emb}} = \gamma_{0,\mathrm{head}}$, the ADAMW curves make comparable or slightly faster initial progress, but remain consistently above the ROWNORMM and HYBRIDPOLARGRADM curves later in training. This qualitative ordering holds both when the SwiGLU MLP projection matrices use MUON and when they use HYBRIDPOLARGRADM. Thus, the improvement from symmetry-compatible vocabulary-indexed updates is robust to reasonable base learning-rate variation and does not depend on simultaneously replacing MUON on the SwiGLU MLP projections.

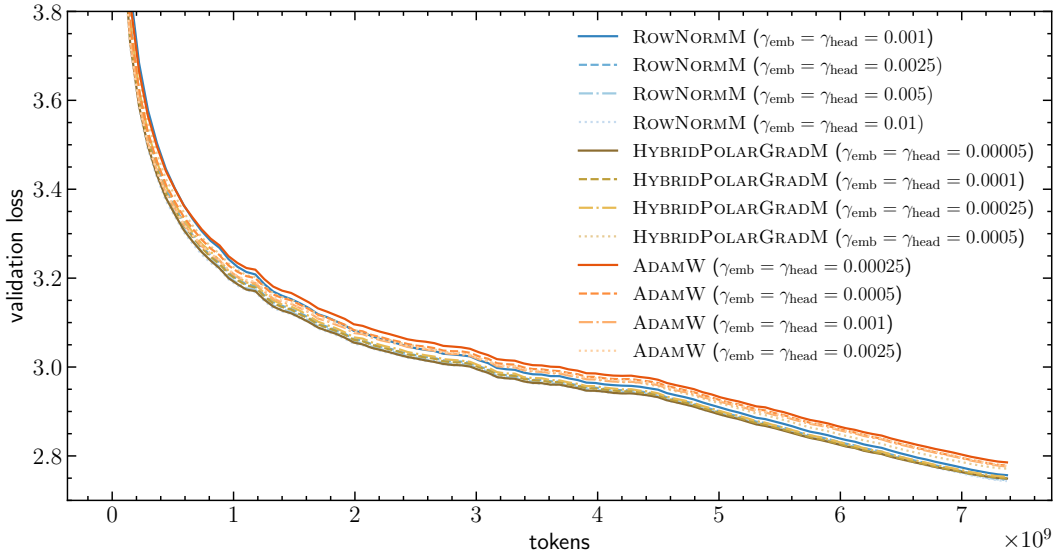
K.1.2 Gemma 3 1B-Style Pre-Training Across Random Seeds

To assess the robustness of the observed optimizer ordering, we repeat the Gemma 3 1B-style pre-training experiment with two additional random seeds. As in Section K.1, we consider the same three optimizer assignments for the input embedding and LM head matrices: (i) ROWNORMM, (ii) HYBRIDPOLARGRADM, and (iii) ADAMW. In all runs in this subsection, the SwiGLU MLP projection matrices use HYBRIDPOLARGRADM with a row-norm/right-spectral composition.

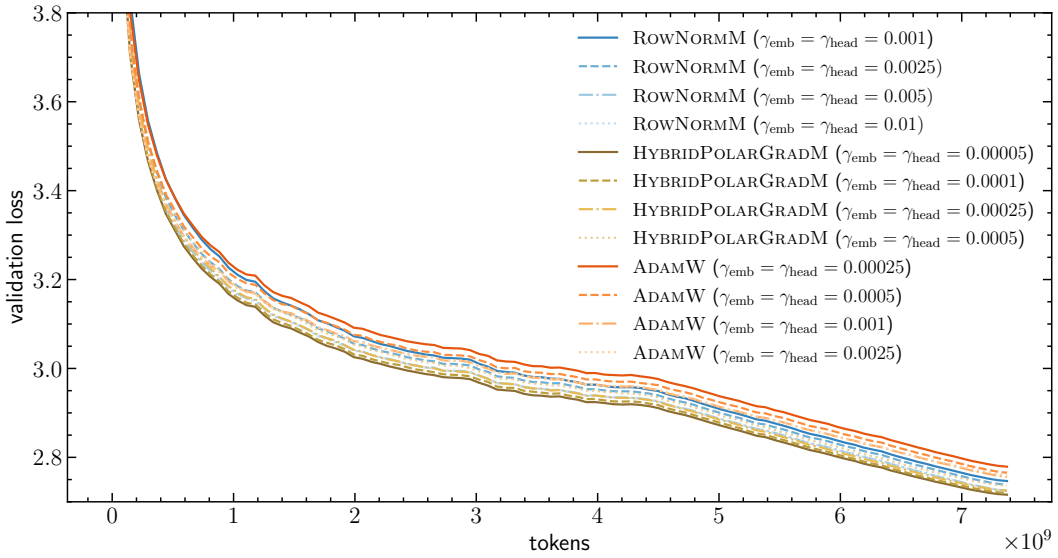
Table 3: Final validation losses across three random seeds for Gemma 3 1B-style pre-training. The optimizer assignment varies only for the input embedding and LM head matrices; SwiGLU MLP projection matrices use HYBRIDPOLARGRADM throughout.

Embedding/LM head optimizer	Seed 1	Seed 2	Seed 3	Mean loss $\pm$ sample std
ROWNORMM	2.7251	2.7267	2.7243	2.7254 ± 0.0012
HYBRIDPOLARGRADM	2.7160	2.7172	2.7166	$\mathbf{2.7166 \pm 0.0006}$
ADAMW	2.7562	2.7612	2.7591	2.7588 ± 0.0025

As shown in Table 3, the qualitative ordering is stable across random seeds. Both symmetry-compatible optimizer assignments, ROWNORMM and HYBRIDPOLARGRADM, consistently outperform the ADAMW baseline on the vocabulary-indexed matrices. HYBRIDPOLARGRADM achieves the lowest mean final validation loss, improving over ADAMW by approximately 0.0422, while



(a) SwiGLU MLP projection matrices use Muon.



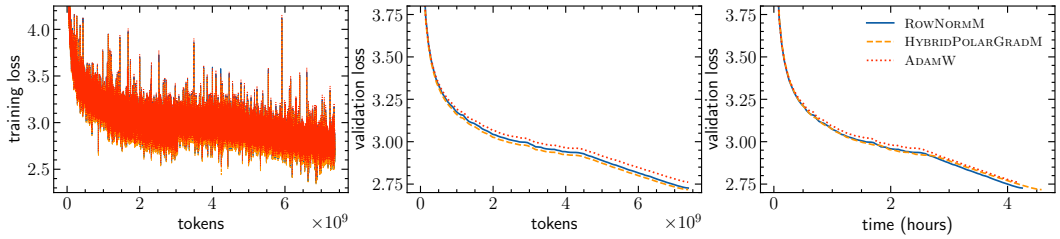
(b) SwiGLU MLP projection matrices use HybridPolarGradM with a row-norm/right-spectral composition.

Figure 5: Validation loss curves for the Gemma 3 1B-style embedding/LM head learning-rate sweep. The swept learning rate is $\gamma_{0,\text{emb}} = \gamma_{0,\text{head}}$. The two panels differ only in the optimizer assigned to the SwiGLU MLP projection matrices: Muon in Figure 5a and HybridPolarGradM in Figure 5b. Across both settings, RowNormM and HybridPolarGradM remain consistently better than AdamW for the input embedding and LM head matrices.

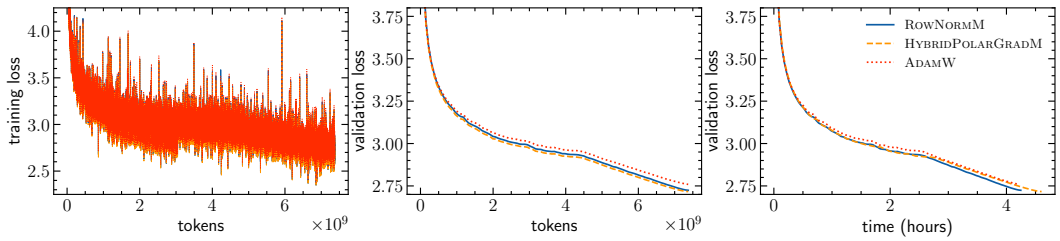
RowNormM improves over AdamW by approximately 0.0335. These results indicate that the gains observed in the main Gemma 3 1B-style experiment are not an artifact of a single random initialization.

K.1.3 Vocabulary-Logit Growth and Final Logit Soft-Capping

We additionally study whether symmetry-compatible LM head updates affect the growth of final vocabulary logits in dense pre-training. This diagnostic is motivated by prior work on training instabilities in large Transformer models. Chowdhery et al. [26] introduce an auxiliary output z-loss,



(a) Second random seed.



(b) Third random seed.

Figure 6: Training and validation losses for Gemma 3 1B-style pre-training under two additional random seeds. In each subfigure, the three configurations differ only in the optimizer assigned to the input embedding and LM head matrices: RowNormM, HybridPolarGradM, or AdamW. The SwiGLU MLP projection matrices use HybridPolarGradM in all runs.

proportional to $\log^2 Z$, to keep the softmax normalizer $\log Z$ close to zero and improve stability. Wortsman et al. [147] study small-scale proxies for large-scale Transformer instabilities, including attention-logit growth and divergence between output logits and log probabilities. Gemma 2 [46] instead uses explicit logit soft-capping in both attention layers and the final vocabulary layer, applying $z \leftarrow c \tanh(z/c)$ with soft-cap values $c = 50$ for attention logits and $c = 30$ for final logits.

Our goal is to test whether optimizer geometry provides a complementary mechanism for controlling final vocabulary logits. For an LM head $W_{\text{out}} \in \mathbb{R}^{v \times d}$, the final logits are $z_t = W_{\text{out}} h_t \in \mathbb{R}^v$. Since the softmax distribution is invariant under shared shifts $z_t \mapsto z_t + c\mathbf{1}_v$, the LM head has a natural shared-logit-shift quotient geometry. The projected LM head updates proposed in this paper remove this symmetry-redundant direction from the actual parameter update, and may therefore reduce unnecessary or unstable growth of the final vocabulary logits.

We run Gemma 3 1B-style pre-training while varying the LM head optimizer and the final-logit soft-capping setting. Attention-logit soft-capping is disabled in all runs, so the diagnostic isolates the final vocabulary logits. In addition to validation cross-entropy, we log final-logit diagnostics on sampled validation token positions. Let $\mathcal{S}$ denote the sampled positions. For each $t \in \mathcal{S}$, let

$$z_t = (z_{t,i})_{1 \leqslant i \leqslant v} = W_{\text{out}} h_t \in \mathbb{R}^v, \qquad \overline{z}_t := \frac{1}{v} \sum_{i=1}^{v} z_{t,i}, \qquad \widetilde{z}_t := z_t - \overline{z}_t \mathbf{1}_v.$$

We report the raw final-logit RMS, centered final-logit RMS, and maximum log-sum-exp given by

$$\left(\frac{1}{|\mathcal{S}|} \sum_{t \in \mathcal{S}} \frac{\|z_t\|_2^2}{v} \right)^{1/2}, \qquad \left(\frac{1}{|\mathcal{S}|} \sum_{t \in \mathcal{S}} \frac{\|\widetilde{z}_t\|_2^2}{v} \right)^{1/2}, \qquad \max_{t \in \mathcal{S}} \log \sum_{i=1}^{v} \exp(z_{t,i}).$$

The raw RMS measures the overall logit scale, including the shared-logit-shift component. The centered RMS removes this shared component and measures logit growth on the LM head quotient space. The maximum log-sum-exp is a spike-sensitive diagnostic for unusually large final vocabulary logits. Under distributed training, this maximum is reduced across devices and therefore corresponds to the maximum over the union of sampled positions across ranks.

This experiment tests whether projected symmetry-compatible LM head updates provide an optimizer-side form of logit control. The quotient-geometry prediction is not that such updates shrink all logit

variation, but that they suppress drift in the shared-logit-shift direction while preserving the centered logits that determine the softmax distribution. Thus, a positive outcome is that, without final logit soft-capping, projected RowNormM or HybridPolarGradM LM head updates reduce raw logit RMS and log-sum-exp growth while maintaining comparable centered-logit diagnostics and comparable or improved validation cross-entropy relative to the corresponding AdamW LM head baseline.

In the figures below, RowNormM and HybridPolarGradM are configurations (b)(i) and (b)(ii) in Table 9, respectively, where the same optimizer is used for both the embedding and LM head matrices. RowNormM + AdamW and HybridPolarGradM + AdamW use the first optimizer for the embedding and AdamW for the LM head. All other optimizer hyperparameters are kept as in the original Gemma 3 1B-style setting given in Table 9.

Without final logit soft-capping. We first consider Gemma 3 1B-style pre-training without final logit soft-capping. As shown in Figure 7, projected RowNormM and HybridPolarGradM LM head updates achieve slightly lower validation loss than the corresponding configurations using AdamW on the LM head. The AdamW LM head variants exhibit substantially larger raw logit RMS, with a pronounced late-stage surge for RowNormM + AdamW. The projected updates keep the raw logit RMS much smaller and avoid a comparable surge. Maximum log-sum-exp follows a different pattern: projected HybridPolarGradM exhibits the largest late-stage values despite maintaining the smallest raw logit RMS.

The centered diagnostics indicate that the surge in RowNormM + AdamW is driven mainly by the shared-logit-shift component: its centered logit RMS increases only modestly and remains below that of projected RowNormM. Moreover, raw and centered logit RMS nearly coincide for both projected methods, whereas the AdamW LM head variants retain a substantial gap. The HybridPolarGradM + AdamW configuration also shows a late-stage rise in centered logit RMS. These observations support the quotient-geometry interpretation: projected LM-head updates suppress drift in a symmetry-redundant direction while slightly improving validation performance, without necessarily reducing extreme-value diagnostics such as maximum log-sum-exp.

With final logit soft-capping. We next repeat the comparison with final logit soft-capping enabled. As shown in Figure 8, the AdamW LM head variants no longer exhibit the pronounced late-stage raw logit RMS growth observed without soft-capping. Their raw logit RMS generally decreases after the initial stage, with intermediate fluctuations, while maximum log-sum-exp remains in a comparable range across methods. The projected RowNormM and HybridPolarGradM updates maintain larger but steadier raw logit RMS for most of training, consistent with a stabilizing effect of soft-capping.

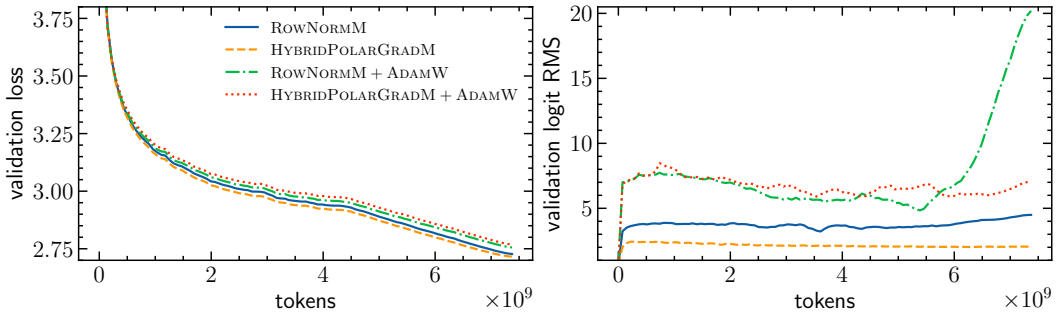
Nevertheless, the projected methods still achieve lower validation loss than the corresponding AdamW LM head variants. Their larger centered logit RMS indicates that this benefit does not simply come from shrinking all logit variation. Raw and centered logit RMS nearly coincide for the projected methods, whereas the AdamW variants retain an appreciable gap. Thus, soft-capping and projected updates exhibit complementary empirical benefits: soft-capping bounds the effective final logits, while the projected configurations maintain smaller shared-shift components and achieve lower validation loss. However, a shared shift of the pre-capping logits is not generally an exact symmetry after nonlinear soft-capping, so the quotient-geometry interpretation requires qualification.

We give all the final validation loss and logit statistics in Table 4.

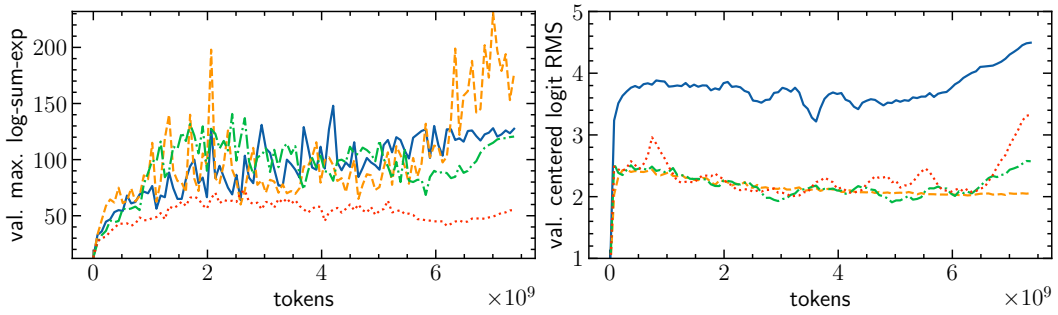
K.1.4 Separation of the Embedding and LM-Head Contributions

To separate the embedding and LM-head contributions, we evaluate mixed optimizer assignments for Gemma 3 1B-style pre-training with untied matrices; see Table 5. Starting from the hyperparameters of the RowNormM/RowNormM and HybridPolarGradM/HybridPolarGradM configurations, we replace either optimizer with AdamW, retaining the original learning rates without retuning. All other training settings remain fixed.

For RowNormM, the embedding-only assignment achieves a validation loss of 2.7497, improving over the AdamW/AdamW reference of 2.7562 by 0.0065. The LM-head-only assignment achieves 2.7275, a larger improvement of 0.0287. Using RowNormM on both matrices reduces the loss to



(a) Validation cross-entropy and raw final-logit RMS.



(b) Validation maximum log-sum-exp and centered final-logit RMS.

Figure 7: Validation loss and final vocabulary-logit diagnostics for Gemma 3 1B-style pre-training without final logit soft-capping. The AdamW LM head variants exhibit larger raw logit RMS and, for RowNormM +AdamW, a late-stage increase in maximum log-sum-exp, while the centered logit RMS remains comparable. This indicates that the growth is mainly in the shared-logit-shift component.

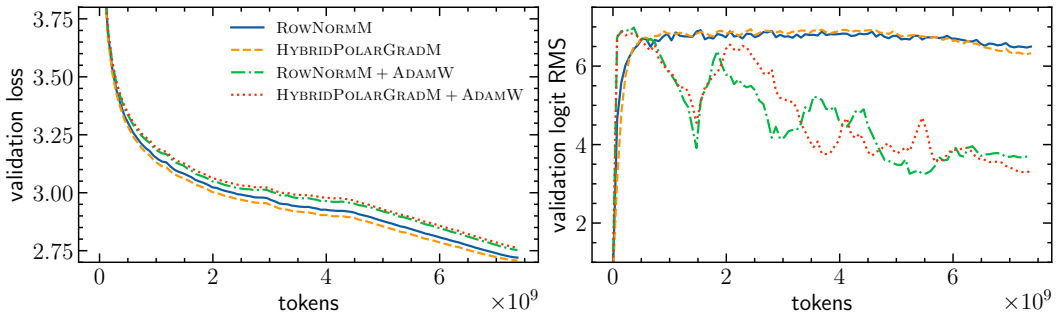
Table 4: Final validation loss and logit statistics for Gemma 3 1B-style pre-training.

Setting	Optimizer	Validation loss	Raw logit RMS	Centered logit RMS	Maximum log-sum-exp
No logit soft-capping	RowNormM	2.7275	4.4944	4.4944	127.50
	HybridPolarGradM	**2.7161**	2.0556	2.0556	177.00
	RowNormM + AdamW	2.7556	20.1984	2.5737	120.57
	HybridPolarGradM + AdamW	2.7685	7.0571	3.3251	55.0087
Logit soft-capping	RowNormM	2.7202	6.4975	6.4971	32.9408
	HybridPolarGradM	**2.7074**	6.3287	6.3285	31.9113
	RowNormM + AdamW	2.7530	3.6942	2.1700	30.8319
	HybridPolarGradM + AdamW	2.7636	3.3124	2.4248	31.5483

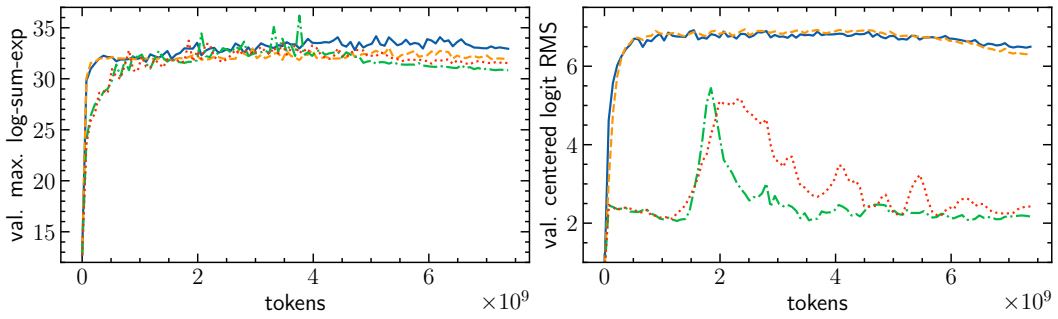
2.7251, giving an overall gain of 0.0311 and an additional embedding contribution of 0.0024 relative to the LM-head-only assignment.

For HybridPolarGradM, the embedding-only assignment increases the loss to 2.8131, whereas the LM-head-only assignment reduces it to 2.7216, a gain of 0.0346 over the reference. The fully hybrid assignment achieves the lowest loss of 2.7160, improving over the reference by 0.0402 and over the LM-head-only assignment by 0.0056.

Under these training recipes, the larger gains arise from changing the LM head optimizer, while the embedding contribution depends on the LM head assignment. In particular, hybrid embedding updates help when paired with a hybrid LM head but perform poorly with an AdamW LM head. These findings are conditional on the inherited hyperparameters, since the mixed assignments were not separately retuned.



(a) Validation cross-entropy and raw final-logit RMS.



(b) Validation maximum log-sum-exp and centered final-logit RMS.

Figure 8: Validation loss and final vocabulary-logit diagnostics for Gemma 3 1B-style pre-training with final logit soft-capping enabled. Soft-capping suppresses the raw logit-scale growth of the AdamW LM head variants, while projected RowNormM and HybridPolarGradM LM head updates still achieve lower validation loss and stronger centered logit variation.

Table 5: Embedding and LM-head optimizer contributions for Gemma 3 1B-style pre-training. Gains denote reductions in validation loss. The LM-head gain compares against an AdamW LM head with the same embedding optimizer.

Embedding optimizer	LM-head optimizer	Validation loss	Gain over AdamW/AdamW	LM-head gain at fixed embedding
AdamW	AdamW	2.7562	N/A	N/A
RowNormM	AdamW	2.7497	0.0065	N/A
AdamW	RowNormM	2.7275	0.0287	0.0287
RowNormM	RowNormM	2.7251	0.0311	0.0246
HybridPolarGradM	AdamW	2.8131	−0.0569	N/A
AdamW	HybridPolarGradM	2.7216	0.0346	0.0346
HybridPolarGradM	HybridPolarGradM	2.7160	0.0402	0.0971

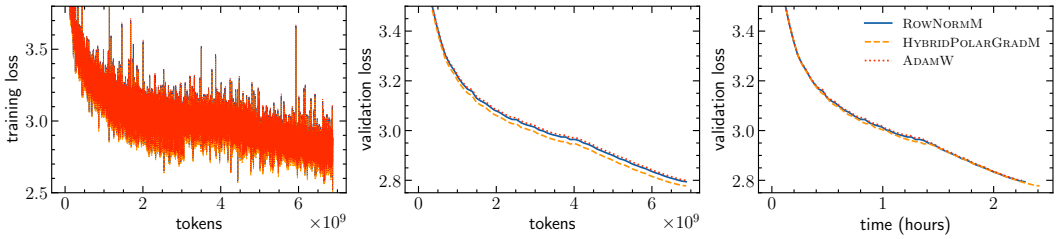
K.2 Qwen3-0.6B-Style Pre-Training

We pre-train a Qwen3-0.6B-style dense language model [119] incorporating several recent architectural innovations, including Grouped Query Attention (GQA) [4], the SwiGLU activation function [28, 127], Rotary Positional Embeddings (RoPE) [138], pre-norm normalization [150] with RMSNorm [71, 163], and QK-Norm [32], without QKV bias terms. The model uses a vocabulary size of 151,936 and hidden dimension 1024. Since untying the embeddings increases the number of trainable parameters, we reduce the number of hidden layers from 28 to 20, resulting in a total of 625,784,832 trainable parameters.

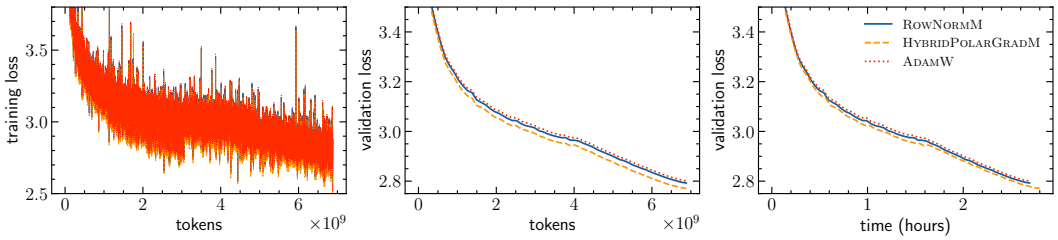
We compare three optimizer assignments for the vocabulary-indexed matrices, namely the input embedding and LM head matrices: (i) RowNormM, (ii) HybridPolarGradM with row-norm/right-spectral, (iii) AdamW. For SwiGLU MLP projections, we compare (a) right-spectral updates,

equivalently MUON-style updates, with (b) hybrid row-norm/right-spectral updates applied along the appropriate intermediate-neuron axis: row-aware for gate and up projections, and column-aware for down projections.

Although different LPRO-compatible optimizers could in principle be assigned to the embedding and LM head matrices, we use the same optimizer for both layers in each configuration for simplicity.



(a) SwiGLU MLP projection matrices use MUON, equivalently RIGHTPOLARGRADM with $\alpha = 0$.



(b) SwiGLU MLP projection matrices use HYBRIDPOLARGRADM with a row-norm/right-spectral composition.

Figure 9: Training and validation losses for Qwen3-0.6B-style pre-training. In each subfigure, the three configurations differ only in the optimizer assigned to the input embedding and LM head matrices: ROWNORMM, HYBRIDPOLARGRADM, or ADAMW.

The final validation losses for configurations (i)–(iii) are 2.7943, 2.7774, and 2.7996 in Figure 9a, and 2.7927, 2.7710, and 2.8009 in Figure 9b, respectively. As shown in Figure 9, in both settings, configuration (iii) makes comparable initial progress to configuration (i) and has lower validation loss at the earlier stage of training, but is subsequently overtaken by both ROWNORMM and HYBRIDPOLARGRADM. The final gap between HYBRIDPOLARGRADM and ADAMW is larger than that between ROWNORMM and ADAMW, and the validation loss favors both symmetry-compatible updates.

Across Figure 9a and Figure 9b, using HYBRIDPOLARGRADM for the SwiGLU MLP projection matrices reduces the final validation loss relative to MUON for the ROWNORMM and HYBRIDPOLARGRADM embedding/LM head configurations, from 2.7943 to 2.7927 and from 2.7774 to 2.7710, respectively. For the ADAMW embedding/LM head configuration, the final validation loss increases slightly from 2.7996 to 2.8009. The improvement is largest when HYBRIDPOLARGRADM is also used for the input embedding and LM head matrices, suggesting a possible complementary effect between symmetry-compatible updates on vocabulary-indexed matrices and row-aware/right-spectral updates on SwiGLU MLP projections. HYBRIDPOLARGRADM remains the best-performing assignment for the input embedding and LM head matrices in both settings. Thus, the comparison between (a) and (b) suggests that applying row-norm/right-spectral hybrid updates to SwiGLU MLP projections can further improve validation loss, while the relative ranking among the three vocabulary-indexed optimizer assignments remains stable. This improvement is plausibly due to the tall-skinny geometry of the SwiGLU gate and up projections. Since $d_{\text{model}} = 1024$ and $d_{\text{ff}} = 3072$, the gate and up projections have many more rows than columns, with rows corresponding to intermediate neurons. Row-scale imbalance can therefore be important. While MUON captures the right-spectral geometry, HYBRIDPOLARGRADM additionally normalizes intermediate-neuron rows before the spectral step, which may yield a better update geometry in this regime.

In terms of wall-clock training time, HYBRIDPOLARGRADM incurs additional overhead due to the inner Gram Newton–Schulz iterations used to approximate the right-spectral component. Consequently, configurations (ii) and (iii) require a similar amount of training time to reach comparable

validation loss values, despite HYBRIDPOLARGRADM achieving a slightly lower final loss. Configurations (i) and (ii) both follow symmetry-compatible geometries for the embedding and LM head matrices, whereas configuration (iii) applies coordinate-wise ADAMW updates to these matrix-valued parameters, thereby introducing a geometry mismatch. Overall, these results are consistent with our symmetry-aware matrix view of optimizer design: even when applied only to the vocabulary-indexed matrices, geometry-compatible updates can improve the optimization trajectory and final validation loss.

K.3 OLMOE-1B-7B-Style Pre-Training

We pre-train AllenAI's OLMOE-1B-7B [107], which provides a comprehensive training recipe together with open-source data, code, and training logs. The model has vocabulary size 50,304 and hidden dimension 2048, making the embedding and LM head matrices considerably large. We also reduce the number of hidden layers from 16 to 12 and the number of experts from 64 to 32, yielding a total of 2,824,177,664 trainable parameters. We use the same loss function and the same four optimizer assignments as in the downsized gpt-oss experiment in Section 5.

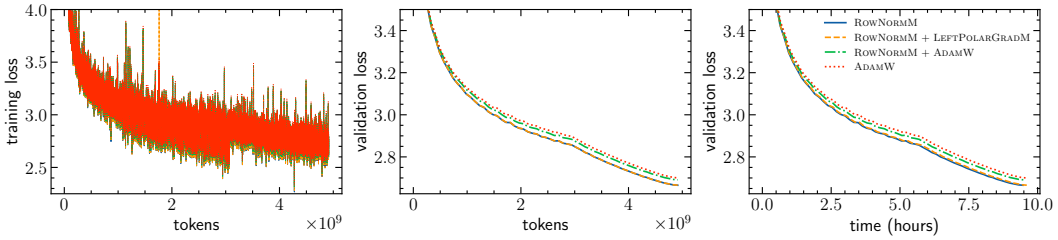


Figure 10: Training and validation losses for OLMOE-1B-7B-style pre-training. The configurations differ in the optimizers assigned to the embedding, LM head, and router matrices.

The final validation losses for configurations (i)–(iv) are 4.3971, 4.4080, 4.4312, and 4.4370, respectively. As shown in Figure 10, configuration (iv), which uses ADAMW for the embedding, LM head, and router matrices, makes faster initial progress over roughly the first 10,000 steps, but is eventually overtaken by configurations (i)–(iii). This reversal occurs before the onset of learning rate decay, which linearly decreases the learning rate to zero from step 18,000 to step 30,000. The validation loss gaps continue to widen during the decay phase, suggesting that the advantage of the geometry-compatible configurations may become more pronounced in longer training runs.

Configurations (i) and (ii) use symmetry-compatible updates for all three special matrix classes considered here: embeddings, LM head, and routers. Configuration (iii) retains symmetry-compatible updates for the embedding and LM head matrices, but introduces a geometry mismatch in the router updates by using coordinate-wise ADAMW. Configuration (iv) applies ADAMW to all three classes and therefore departs most strongly from the proposed symmetry-compatible optimizer design. Empirically, both (i) and (ii) outperform (iii), while (iv) performs worst overall. These results are consistent with our theoretical perspective that respecting parameter symmetry and matrix geometry can matter for optimizer design, especially in large sparse architectures where router dynamics play a central role.

The performance gap between configurations (i) and (ii) is relatively small, suggesting that the row-norm and left-spectral router updates behave similarly in this setting. This small gap may reflect suboptimal learning-rate tuning for LEFTPOLARGRADM, the effect of inexact polar oracles, or simply that row-wise normalization already captures much of the relevant router geometry in this experiment. Finally, we observe that configuration (iv) exhibits two more pronounced training-loss spikes between steps 13,000 and 18,000 than the other configurations, despite using MUON for the hidden and attention matrices. This suggests that geometry-matched optimizer choices for embeddings, LM heads, and routers may also improve training stability in practice.

K.4 Load-Balancing Loss and Router z-Loss Diagnostics for Downsized gpt-oss Pre-Training

Following the diagnostic protocol of [107], we evaluate whether symmetry-compatible router updates improve expert load balancing and router stability. For each MOE layer, we log the fraction of routed

assignments sent to each expert, the total router probability mass assigned to each expert, the auxiliary load-balancing loss [129], and the router z-loss [173]. We also track load entropy, load coefficient of variation, dead-expert fraction, maximum expert load, and router-logit scale. These diagnostics allow us to distinguish improvements in language model validation loss from improvements in routing dynamics.

Let E denote the number of experts and let k denote the number of activated experts per token. We measure the router load-balancing loss by $\mathcal{L}_{\mathrm{LB}} = E \sum_{i=1}^{E} f_i P_i$, where f_i is the fraction of routed top-k assignments sent to expert i, and P_i is the average router probability mass assigned to expert i. A perfectly balanced router has $\mathcal{L}_{\mathrm{LB}}$ close to 1. We also measure the router z-loss $\mathcal{L}_{\mathrm{z}} = \frac{1}{N} \sum_{t=1}^{N} \left(\log \sum_{i=1}^{E} \exp(z_{t,i}) \right)^2$, where $z_{t,i}$ is the router logit for token t and expert i and N is the batch size. This term penalizes excessive router-logit scale.

We compare RowNormM, LeftPolarGradM, uncentered LeftPolarGradM, HybridPolarGradM, and AdamW on router matrices while keeping all non-router optimizer assignments fixed: AdamW for scalars and vectors, Muon for linear, attention, and SwiGLU MLP matrices, and RowNormM for embedding and LM head matrices. The uncentered LeftPolarGradM baseline applies a Muon-style left-polar update directly to the router momentum without removing the shared-row component, and therefore ablates the centering operation induced by shared-logit-shift invariance. We keep the settings from Tables 14 and 15. The hyperparameters of uncentered LeftPolarGradM are set to be the same as those of RowNormM and LeftPolarGradM, while HybridPolarGradM uses a smaller learning rate 0.00005.

Training objective without auxiliary losses. We first evaluate router dynamics when neither $\mathcal{L}_{\mathrm{LB}}$ nor $\mathcal{L}_{\mathrm{z}}$ is included in the training objective. As shown in Figures 11 and 12, the choice of router optimizer has a large effect on both expert load balancing and router-logit scale. Compared with AdamW, all four alternative router updates achieve substantially lower measured load-balancing loss and router z-loss on both training and validation batches. The centered and uncentered LeftPolarGradM variants give the strongest load-balancing performance, while HybridPolarGradM achieves the lowest router z-loss. On the load-balancing diagnostics, HybridPolarGradM improves over RowNormM but remains less effective than the purely left-polar variants.

The expert-assignment summaries in Figure 13 show a similar pattern. AdamW has lower load entropy, higher load coefficient of variation, a larger dead-expert fraction, and higher maximum expert load. In contrast, both LeftPolarGradM variants achieve the most balanced expert assignment and report zero dead-expert fractions at the final evaluation. HybridPolarGradM is intermediate between the left-polar variants and RowNormM, with a dead-expert fraction of 0.0052, compared with 0.0156 for RowNormM and 0.2396 for AdamW. These results suggest that matrix-aware router updates can improve load balancing in this setting, even without explicitly optimizing a load-balancing auxiliary objective.

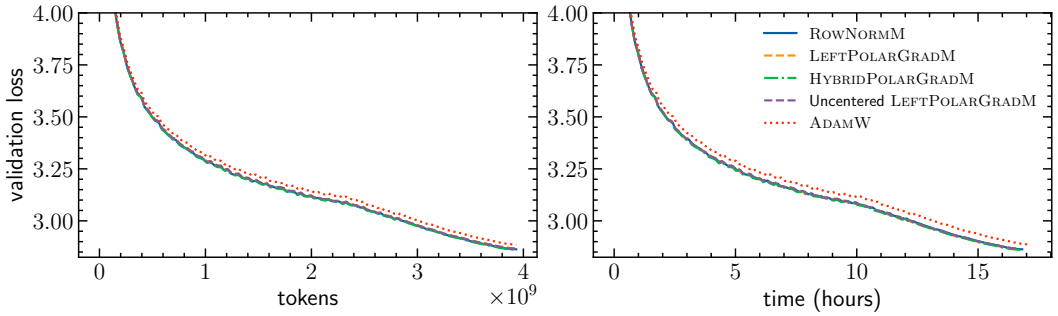
Validation cross-entropy also improves over the AdamW router baseline. HybridPolarGradM achieves the lowest final loss of 2.8606, followed by RowNormM at 2.8631. Centered and uncentered LeftPolarGradM achieve 2.8670 and 2.8666, respectively, compared with 2.8858 for AdamW. Thus, the strongest load balancing does not coincide with the lowest validation loss. The centered and uncentered LeftPolarGradM variants behave similarly, suggesting that their observed gains arise mainly from the left-polar matrix geometry, while centering provides the symmetry-compatible quotient-space formulation induced by shared-logit-shift invariance.

Training objective with auxiliary losses. We next include $\mathcal{L}_{\mathrm{LB}}$ and $\mathcal{L}_{\mathrm{z}}$ in the training objective with weights 0.01 and 0.001, respectively, following the choices of similar work [107, 131, 153]. In this setting, we report validation cross-entropy as the primary language modeling metric, since the total training objective additionally contains auxiliary router penalties. As shown in Figures 14 and 15, auxiliary losses improve routing diagnostics for all methods: the measured load-balancing loss moves closer to its balanced reference value of 1, and the router z-loss decreases substantially.

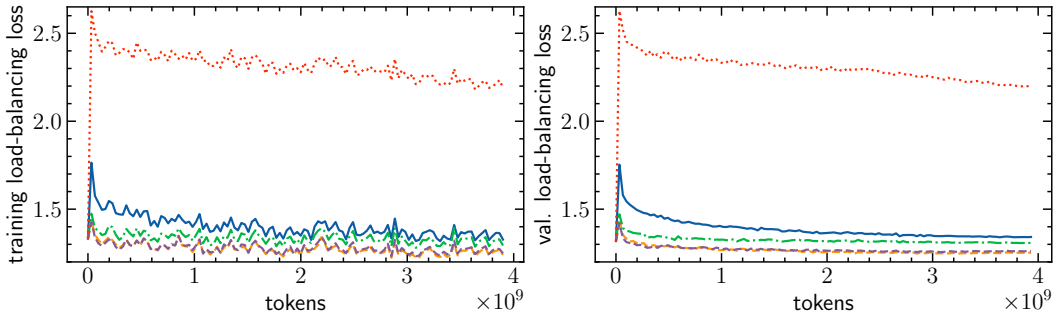
Nevertheless, optimizer geometry remains important. Compared with AdamW, all four alternative router updates yield lower measured load-balancing loss, lower router z-loss, higher load entropy, lower load coefficient of variation, and lower maximum expert load; see Figure 16. All methods, including AdamW, report zero dead-expert fractions at the final evaluation. The four alternatives

are closely clustered, with HYBRIDPOLARGRADM achieving the lowest validation load-balancing loss and router z-loss, at 1.0260 and 0.1376, respectively. Uncentered LEFTPOLARGRADM has the highest load entropy and the lowest load coefficient of variation and maximum expert load.

ROWNORMM achieves the lowest final validation cross-entropy of 2.8605, closely followed by HYBRIDPOLARGRADM at 2.8612. Centered and uncentered LEFTPOLARGRADM achieve 2.8659 and 2.8658, respectively, compared with 2.8693 for ADAMW. Thus, auxiliary router losses and matrix-aware router updates provide complementary improvements in routing diagnostics. These improvements do not uniformly translate into lower validation cross-entropy, however: HYBRIDPOLARGRADM changes slightly from 2.8606 without auxiliary losses to 2.8612 with them.



(a) Validation cross-entropy loss.



(b) Training and validation load-balancing loss.

Figure 11: Validation loss and measured load-balancing loss for downsized gpt-oss pre-training. The configurations differ only in the optimizer assigned to the MOE router matrices. Lower load-balancing loss indicates more balanced expert usage.

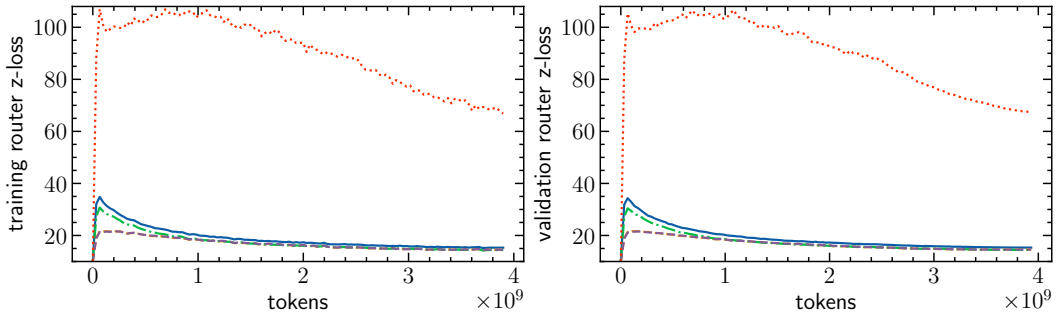
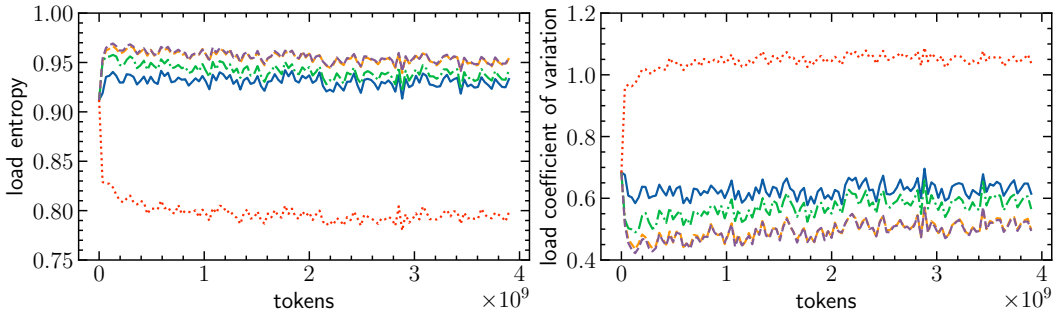
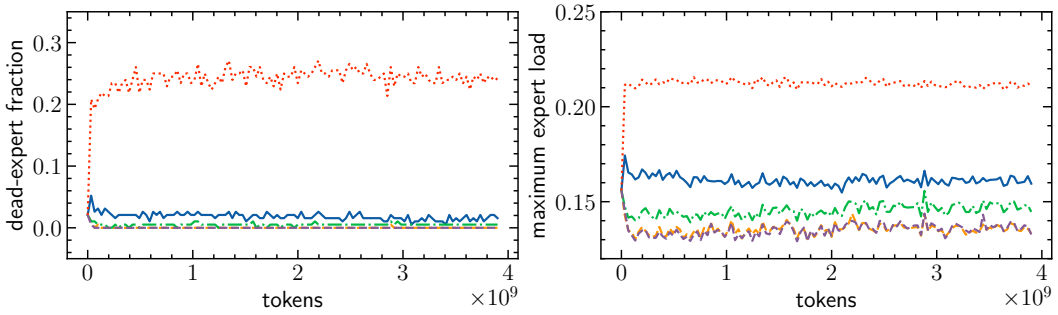


Figure 12: Training and validation router z-loss for downsized gpt-oss pre-training. Lower router z-loss indicates better control of router-logit scale.

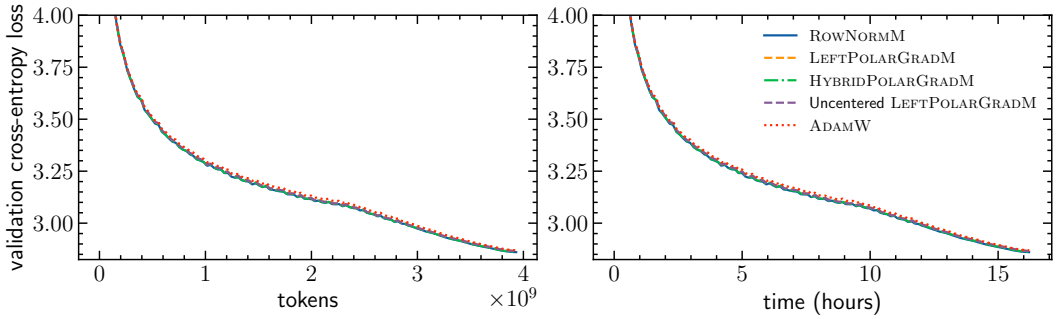


(a) Load entropy and load coefficient of variation. Higher entropy and lower coefficient of variation indicate more balanced expert assignment.

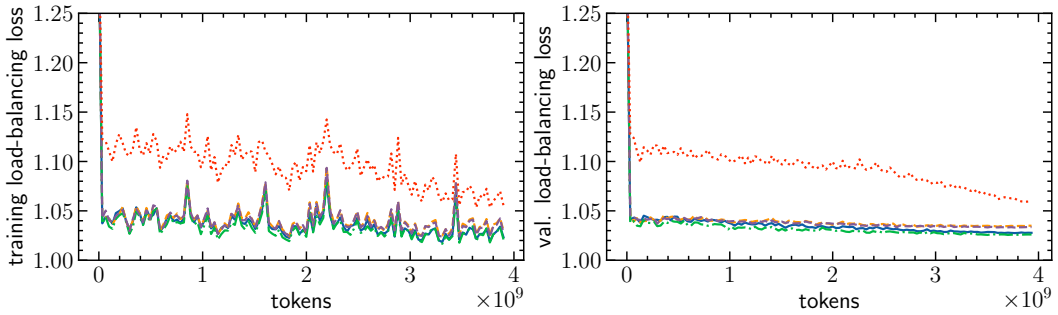


(b) Dead-expert fraction and maximum expert load. Lower values indicate fewer unused experts and less expert dominance.

Figure 13: Expert-assignment balance summaries for downsized gpt-oss pre-training. These diagnostics directly measure whether routed assignments are evenly distributed across experts.



(a) Validation cross-entropy loss.



(b) Training and validation load-balancing loss.

Figure 14: Validation loss and measured load-balancing loss for downsized gpt-oss pre-training with auxiliary router losses enabled. We report validation cross-entropy as the primary language modeling metric because the total training objective also includes auxiliary router losses.

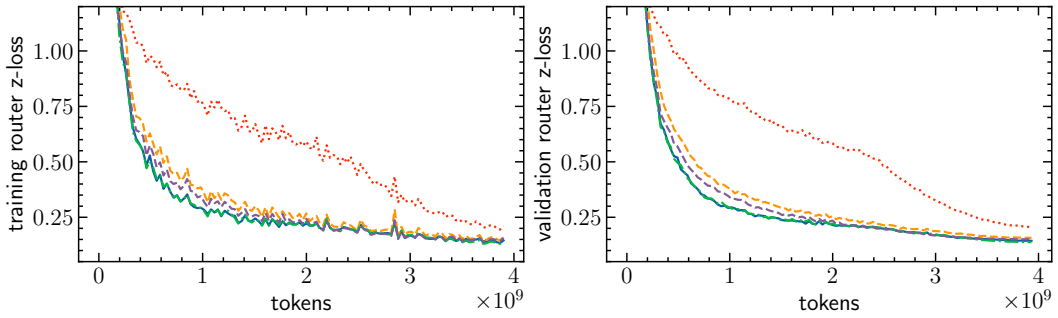
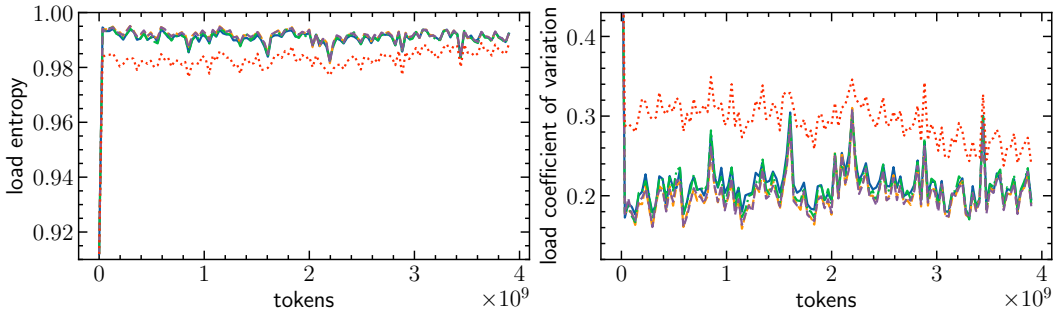
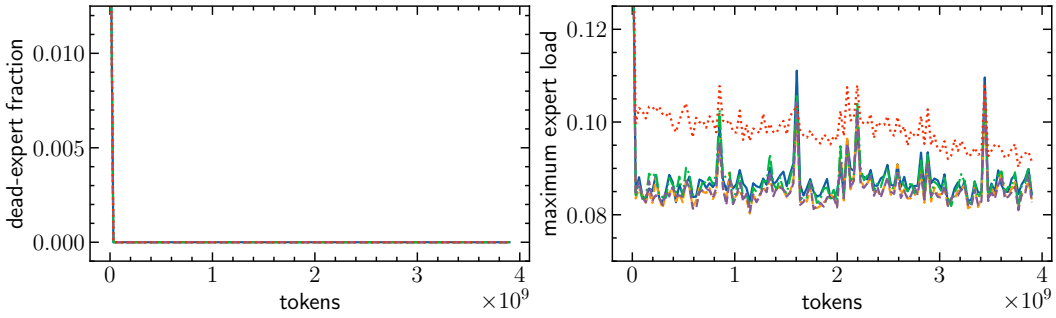


Figure 15: Training and validation router z-loss for downsized gpt-oss pre-training with auxiliary router losses enabled. Lower router z-loss indicates better control of router-logit scale.



(a) Load entropy and load coefficient of variation. Higher entropy and lower coefficient of variation indicate more balanced expert assignment.



(b) Dead-expert fraction and maximum expert load. Lower values indicate fewer unused experts and less expert dominance.

Figure 16: Expert-assignment balance summaries for downsized gpt-oss pre-training with auxiliary router losses enabled. These diagnostics directly measure whether routed assignments are evenly distributed across experts.

We give all the final validation loss and router statistics in Table 6.

Table 6: Final validation loss and router statistics for downsized gpt-oss pre-training, with and without the auxiliary load-balancing and router z-losses. Higher load entropy indicates more balanced expert utilization, whereas lower values are preferable for the other router statistics. The best value within each setting is shown in bold.

Optimizer	Validation loss	Load-balancing loss		Router z-loss	
		Train	Validation	Train	Validation
Without auxiliary losses					
RowNormM	2.8631	1.3269	1.3418	15.3222	15.3382
LeftPolarGradM	2.8670	**1.2401**	**1.2527**	14.6206	14.5886
HybridPolarGradM	**2.8606**	1.2934	1.3082	**14.4418**	**14.4332**
Uncentered LeftPolarGradM	2.8666	1.2493	1.2614	14.6633	14.5773
AdamW	2.8858	2.1903	2.2006	66.8603	67.5009
With auxiliary losses					
RowNormM	**2.8605**	1.0234	1.0278	0.1457	0.1438
LeftPolarGradM	2.8659	1.0298	1.0349	0.1645	0.1572
HybridPolarGradM	2.8612	**1.0210**	**1.0260**	**0.1421**	**0.1376**
Uncentered LeftPolarGradM	2.8658	1.0274	1.0336	0.1587	0.1507
AdamW	2.8693	1.0553	1.0596	0.2107	0.2064

Optimizer	Load entropy	Load CV	Dead-expert frac.	Max. expert load
Without auxiliary losses				
RowNormM	0.9332	0.6148	0.0156	0.1595
LeftPolarGradM	0.9545	0.4971	**0.0000**	0.1334
HybridPolarGradM	0.9424	0.5650	0.0052	0.1446
Uncentered LeftPolarGradM	**0.9549**	**0.4940**	**0.0000**	**0.1327**
AdamW	0.7987	1.0373	0.2396	0.2110
With auxiliary losses				
RowNormM	0.9923	0.1954	**0.0000**	0.0844
LeftPolarGradM	0.9924	0.1895	**0.0000**	0.0833
HybridPolarGradM	0.9926	0.1883	**0.0000**	0.0846
Uncentered LeftPolarGradM	**0.9927**	**0.1868**	**0.0000**	**0.0827**
AdamW	0.9885	0.2398	**0.0000**	0.0915

K.5 Cross-Model Comparison

We finally compare the results across Sections 5 and K.1 to K.3. These experiments span dense and sparse language models with increasing numbers of trainable parameters, from the Qwen3-0.6B-style dense model to the downsized gpt-oss sparse MoE model. Across these models, the vocabulary sizes differ substantially, while the hidden dimensions are relatively close. Consequently, the embedding and LM head matrices have comparable column dimensions, determined by d_{model}, but very different numbers of rows, determined by the vocabulary size v. Thus, as v grows, these vocabulary-indexed matrices become increasingly tall. This changes both their computational cost and the conditioning properties of their matrix gradients, and makes them a natural testbed for row-aware and one-sided spectral optimizer design.

The same row-versus-column distinction is also important for SwiGLU MLP projection matrices. For a dense SwiGLU block, the gate and up projections have shape $d_{\text{ff}} \times d_{\text{model}}$, so their rows correspond to intermediate neurons, whereas the down projection has shape $d_{\text{model}} \times d_{\text{ff}}$, so the same intermediate-neuron geometry appears along the column axis. In the dense Qwen3-0.6B-style and Gemma 3 1B-style models, d_{ff} is substantially larger than d_{model}, placing the gate and up projections in a tall-skinny regime. This is precisely the setting where row-normalization before a right-spectral step can be useful: HybridPolarGradM can correct row-scale imbalance across intermediate neurons while retaining spectral geometry. In the MoE experiments, by contrast, the hidden and

intermediate dimensions are closer in our downsized settings, so a pure MUON-style right-spectral update may already capture much of the relevant matrix geometry for expert SwiGLU projection tensors.

Across all experiments, replacing ADAMW on the large vocabulary-indexed matrices with symmetry-compatible optimizers consistently improves final validation loss. The gains are modest but visible for the smaller dense model, become more pronounced in the larger Gemma 3 1B-style experiment, and persist in both sparse MOE experiments. This trend is consistent with our matrix-geometry perspective: as embedding and LM head matrices grow in their row dimension, coordinate-wise updates increasingly operate in a parameterization-dependent augmented space, whereas row-norm and spectral updates preserve the relevant vocabulary-indexed matrix geometry.

The comparison should not be interpreted as a scaling law, since the models differ in architecture, vocabulary size, training length, and sparsity structure. Nevertheless, the consistent ordering across dense and sparse MOE models provides evidence that the benefit of symmetry-compatible optimizer assignments is not restricted to a single model family or architecture. In particular, the results suggest that large vocabulary-indexed matrices are a robust setting in which geometry-compatible updates can improve optimization. The sparse MOE experiments further show that router matrices and expert SwiGLU projection tensors provide additional layer types where symmetry-aware optimizer design can matter.

Taken together, these experiments support the view that symmetry-compatible optimizer design is most naturally applied as a layerwise optimizer stack, rather than as a single global replacement for ADAMW.

L Experimental Details

Note that we reduce the number of hidden layers and the number of experts from the original architecture. We give the main modified model architecture specifications of the model in Table 7. Their detailed designs can be found in their corresponding technical reports [47, 107, 112, 119]. We initialize all 2D weights by Gaussian random numbers with zero mean and standard deviation 0.02.

Table 7: Modified model architectures of language models.

Model	Qwen3-0.6B	Gemma 3 1B	OLMOE-1B-7B	gpt-oss
# trainable parameters	625,784,832	1,087,138,944	2,824,177,664	3,467,779,008
d_{model}	1024	1152	2048	2048
d_{ff}	3072	6912	1024	2048
n_{layers}	20	18	12	12
n_{heads} (Q / KV)	16 / 8	4 / 1	16 / 16	64 / 8
d_{heads}	128	256	128	64
n_{experts}	1	1	32	16
n_{experts} activated	N/A	N/A	8	4
vocabulary size	151,936	262,144	50,304	201,088
layer norm	RMSNorm	RMSNorm	RMSNorm	RMSNorm
activation function	SwiGLU	GeGLU with tanh	SwiGLU	SwiGLU[2]

In the following experiments, we use POLAR EXPRESS [5] for computing the matrix inverse square root in LEFTPOLARGRADM and Gram Newton–Schulz [164] for computing the orthogonal polar factor directly in HYBRIDPOLARGRADM, respectively. Unless otherwise specified, for POLAR EXPRESS and Gram Newton–Schulz, we use 5 inner steps with $\varepsilon_{\text{NS}} = 10^{-7}$. We use $\varepsilon = 10^{-8}$ for all of ROWNORMM, LEFTPOLARGRADM and HYBRIDPOLARGRADM. The row-scaling rule in ROWNORMM and HYBRIDPOLARGRADM is chosen to be the *smoothed row normalization*, i.e., $\eta(t) = 1/(t + \varepsilon)$ with $\varepsilon = 10^{-8}$.

For ADAMW on scalar and vector parameters, we use a linear warmup with cosine decay learning rate schedule with 100 warmup steps and a half-cosine decay to 0. For all other parameters, regardless of the choice of the optimizers, we use a stable-decay schedule with an initial learning rate γ_0

[2]The SwiGLU implementation in gpt-oss is unconventional as it includes clamping and residual connection.

for the first 60% of training steps and linear decay to 0 for the last 40% training steps. We use the fused implementation of AdamW in PyTorch, while the implementations of RowNormM, LeftPolarGradM, RightPolarGradM and HybridPolarGradM are not optimized with customized kernels, except for the usage of Gram Newton–Schulz [164].

We give the training configurations and optimizer hyperparameters of all four model pre-training experiments in the following subsections.

L.1 Gemma 3 1B-Style Pre-Training

In terms of wall-clock training time, for (a), configurations (i)–(iii) take 3.749 hours, 4.050 hours and 3.682 hours respectively, whereas for (b), configurations (i)–(iii) take 4.216 hours, 4.566 hours and 4.191 hours respectively.

Table 8: Training configurations for Gemma 3 1B-style pre-training.

Model	Gemma 3 1B on FineWeb-Edu-10B
Context length	1024 tokens
Per-device batch size	18 sequences
Training steps	50,000
Training tokens	7,372,800,000
Validation steps	71
Validation tokens	10,469,376
Precision	`bfloat16`
Data-parallel size	8 (Nvidia H200)

Table 9: Optimizer hyperparameters for Gemma 3 1B-style pre-training.

Configuration	Parameter type	Optimizer	γ_0	momentum β	weight decay λ
(a)(i) AdamW + Muon + Muon + RowNormM	scalar / vector	AdamW	0.05	$(0.9, 0.95)$	0.001
	linear / attention	Muon	0.02	0.95	0.001
	SwiGLU MLP	Muon	0.02	0.95	0.001
	embedding	RowNormM	0.01	0.95	0
	head	RowNormM	0.01	0.95	0
(a)(ii) AdamW + Muon + Muon + HybridPolarGradM (row-norm/right-spectral; $\alpha = 1$)	scalar / vector	AdamW	0.05	$(0.9, 0.95)$	0.001
	linear / attention	Muon	0.02	0.95	0.001
	SwiGLU MLP	Muon	0.02	0.95	0.001
	embedding	HybridPolarGradM	0.00005	0.95	0
	head	HybridPolarGradM	0.00005	0.95	0
(a)(iii) AdamW + Muon + Muon + AdamW	scalar / vector	AdamW	0.05	$(0.9, 0.95)$	0.001
	linear / attention	Muon	0.02	0.95	0.001
	SwiGLU MLP	Muon	0.02	0.95	0.001
	embedding	AdamW	0.001	$(0.9, 0.95)$	0
	head	AdamW	0.001	$(0.9, 0.95)$	0
(b)(i) AdamW + Muon + HybridPolarGradM (row-norm/right-spectral; $\alpha = 0$) + RowNormM	scalar / vector	AdamW	0.05	$(0.9, 0.95)$	0.001
	linear / attention	Muon	0.02	0.95	0.001
	SwiGLU gate/up	HybridPolarGradM	$0.02\sqrt{6}$	0.95	0.001
	SwiGLU down	HybridPolarGradM	0.02	0.95	0.001
	embedding	RowNormM	0.005	0.95	0
	head	RowNormM	0.005	0.95	0
(b)(ii) AdamW + Muon + HybridPolarGradM + HybridPolarGradM (row-norm/right-spectral) (MLP: $\alpha = 0$; embedding/head: $\alpha = 1$)	scalar / vector	AdamW	0.05	$(0.9, 0.95)$	0.001
	linear / attention	Muon	0.02	0.95	0.001
	SwiGLU gate/up	HybridPolarGradM	$0.02\sqrt{6}$	0.95	0.001
	SwiGLU down	HybridPolarGradM	0.02	0.95	0.001
	embedding	HybridPolarGradM	0.00005	0.95	0
	head	HybridPolarGradM	0.00005	0.95	0
(b)(iii) AdamW + Muon + HybridPolarGradM (row-norm/right-spectral; $\alpha = 0$) + AdamW	scalar / vector	AdamW	0.05	$(0.9, 0.95)$	0.001
	linear / attention	Muon	0.02	0.95	0.001
	SwiGLU gate/up	HybridPolarGradM	$0.02\sqrt{6}$	0.95	0.001
	SwiGLU down	HybridPolarGradM	0.02	0.95	0.001
	embedding	AdamW	0.001	$(0.9, 0.95)$	0
	head	AdamW	0.001	$(0.9, 0.95)$	0

L.2 Qwen3-0.6B-Style Pre-Training

In terms of wall-clock training time, configurations (i)–(iii) take 7.359 hours, 7.52 hours and 7.38 hours respectively.

Table 10: Training configurations for Qwen3-0.6B-style pre-training.

Model	Qwen3-0.6B on FineWeb-Edu-10B
Context length	1024 tokens
Per-device batch size	28 sequences
Training steps	30,000
Training tokens	6,881,280,000
Validation steps	46
Validation tokens	10,551,296
Precision	`bfloat16`
Data-parallel size	8 (Nvidia H200)

Table 11: Optimizer hyperparameters for Qwen3-0.6B-style pre-training.

Configuration	Parameter type	Optimizer	γ_0	momentum β	weight decay λ
(i) AdamW + Muon + RowNormM	scalar / vector	AdamW	0.05	$(0.9, 0.95)$	0.001
	matrix	Muon	0.02	0.95	0.001
	embedding	RowNormM	0.50	0.95	0
	head	RowNormM	0.005	0.95	0
(ii) AdamW + Muon + HybridPolarGradM (row-norm/right-spectral; $\alpha = 1$)	scalar / vector	AdamW	0.05	$(0.9, 0.95)$	0.001
	matrix	Muon	0.02	0.95	0.001
	embedding	HybridPolarGradM	1.00	0.95	0
	head	HybridPolarGradM	0.01	0.95	0
(iii) AdamW + Muon + AdamW	scalar / vector	AdamW	0.05	$(0.9, 0.95)$	0.001
	matrix	Muon	0.02	0.95	0.001
	embedding	AdamW	0.10	$(0.9, 0.95)$	0
	head	AdamW	0.001	$(0.9, 0.95)$	0

L.3 OLMoE-1B-7B-Style Pre-Training

In terms of wall-clock training time, configurations (i)–(iv) take 7.907 hours, 7.924 hours, 7.907 hours and 7.908 hours respectively.

Table 12: Training configurations for OLMoE-1B-7B-style pre-training.

Model	OLMoE-1B-7B on FineWeb-Edu-10B
Context length	1024 tokens
Per-device batch size	20 sequences
Training steps	30,000
Training tokens	4,915,200,000
Validation steps	64
Validation tokens	10,485,760
Precision	`bfloat16`
Data-parallel size	8 (Nvidia H200)

Table 13: Optimizer hyperparameters for OLMoE-1B-7B-style pre-training.

Configuration	Parameter type	Optimizer	γ_0	momentum β	weight decay λ
(i) AdamW + Muon + RowNormM	scalar / vector	AdamW	0.01	$(0.9, 0.95)$	0.001
	matrix	Muon	0.005	0.95	0.001
	embedding	RowNormM	0.5	0.95	0
	head	RowNormM	0.005	0.95	0
	router	RowNormM	0.00075	0.95	0
(ii) AdamW + Muon + RowNormM + LeftPolarGradM ($\alpha = 1$)	scalar / vector	AdamW	0.01	$(0.9, 0.95)$	0.001
	matrix	Muon	0.005	0.95	0.001
	embedding	RowNormM	0.5	0.95	0
	head	RowNormM	0.005	0.95	0
	router	LeftPolarGradM	0.00075	0.95	0
(iii) AdamW + Muon + RowNormM + AdamW	scalar / vector	AdamW	0.01	$(0.9, 0.95)$	0.001
	matrix	Muon	0.005	0.95	0.001
	embedding	RowNormM	0.5	0.95	0
	head	RowNormM	0.005	0.95	0
	router	AdamW	0.00075	0.95	0
(iv) AdamW + Muon + AdamW	scalar / vector	AdamW	0.01	$(0.9, 0.95)$	0.001
	matrix	Muon	0.005	0.95	0.001
	embedding	AdamW	0.5	$(0.9, 0.95)$	0
	head	AdamW	0.005	$(0.9, 0.95)$	0
	router	AdamW	0.00075	$(0.9, 0.95)$	0

L.4 Downsized gpt-oss Pre-Training

In terms of wall-clock training time, configurations (i)–(iv) take 14.15 hours, 13.82 hours, 14.2 hours and 14.25 hours respectively.

Table 14: Training configurations for downsized gpt-oss pre-training.

Model	Downsized gpt-oss on FineWeb-Edu-10B
Context length	1024 tokens
Per-device batch size	8 sequences
Training steps	60,000
Training tokens	3,932,160,000
Validation steps	160
Validation tokens	10,485,760
Precision	`bfloat16`
Data-parallel size	8 (Nvidia H200)

Table 15: Optimizer hyperparameters for downsized gpt-oss pre-training.

Configuration	Parameter type	Optimizer	γ_0	momentum β	weight decay λ
(i) AdamW + Muon + RowNormM	scalar / vector	AdamW	0.005	$(0.9, 0.95)$	0.001
	matrix	Muon	0.001	0.95	0.001
	embedding	RowNormM	0.1	0.95	0
	head	RowNormM	0.001	0.95	0
	router	RowNormM	0.00075	0.95	0
(ii) AdamW + Muon + RowNormM + LeftPolarGradM ($\alpha = 1$)	scalar / vector	AdamW	0.005	$(0.9, 0.95)$	0.001
	matrix	Muon	0.001	0.95	0.001
	embedding	RowNormM	0.1	0.95	0
	head	RowNormM	0.001	0.95	0
	router	LeftPolarGradM	0.00075	0.95	0
(iii) AdamW + Muon + RowNormM + AdamW	scalar / vector	AdamW	0.005	$(0.9, 0.95)$	0.001
	matrix	Muon	0.001	0.95	0.001
	embedding	RowNormM	0.1	0.95	0
	head	RowNormM	0.001	0.95	0
	router	AdamW	0.00075	0.95	0
(iv) AdamW + Muon + AdamW	scalar / vector	AdamW	0.005	$(0.9, 0.95)$	0.001
	matrix	Muon	0.001	0.95	0.001
	embedding	AdamW	0.1	$(0.9, 0.95)$	0
	head	AdamW	0.001	$(0.9, 0.95)$	0
	router	AdamW	0.00075	$(0.9, 0.95)$	0

M Prospect of Deep Learning Optimization

All popular coordinate-wise adaptive gradient optimizers are, at their core, geometry-blind: they treat matrix- and tensor-valued parameters as collections of independent coordinates rather than objects with intrinsic symmetry and structure. The developments in this work suggest a different future. Through the principles of symmetry and equivariance, one may derive optimizer geometries that are better aligned with the spaces in which modern deep learning parameters naturally live. This perspective points toward a broader reconsideration of optimizer design in large-scale deep learning, spanning layerwise adaptation, architecture–optimizer co-design, training stabilization, numerical linear algebra, and distributed scalability.

M.1 Emergence of Geometry-Aware Layerwise Optimizers and Architecture–Optimizer Co-Design

Layerwise training is by no means a new idea. Classical examples include layerwise optimizers such as LARS and LAMB [158, 159], which were developed for large-batch training, as well as layerwise hyperparameter prescriptions such as the layerwise learning-rate scaling rules arising in μP [155]. More broadly, the very structure of deep neural network training already necessitates parameter updates in a parallelized Jacobi-style block-coordinate fashion across layers; see e.g., [82, 162].

What is new, however, is the emergence of *geometry-aware* layerwise optimizers: methods that not only act layerwise, but also account for the specific symmetry and matrix geometry of each layer. Recent optimizers such as SHAMPOO [59], MUON [73], SOAP [144], SCION [113], GLUON [124], POLARGRAD [83], and related matrix-gradient methods suggest that modern optimizer design is already moving in this direction. Our development strengthens this trend by proposing a systematic symmetry-based foundation for architecture–optimizer co-design. In particular, it suggests that different parameter classes—such as linear layers, embedding layers, and layers with fused weights such as multi-head attention and grouped query attention [4]—should in principle employ different update geometries matched to their underlying invariance structures.

This perspective becomes even more compelling in modern generative paradigms and foundation models, whose architectures are increasingly heterogeneous. Large language models [18, 121, 122, 142], vision transformers [37], multimodal models (e.g. vision language models [17, 123]), diffusion language models [52, 102, 111], and MOEs [84, 129] all contain parameter blocks with markedly different algebraic and semantic roles, and hence with different natural symmetry classes. Dense linear layers admit bi-orthogonal symmetry; embedding and LM head matrices admit left-permutation and right-orthogonal (and and shared-logit-shift) symmetry; and MOE routers admit expert-permutation and shared-logit-shift symmetry. From this viewpoint, it is increasingly unnatural to optimize all such layers with a single coordinate-wise rule. Rather, these developments point toward a broader principle of *architecture–optimizer co-design*: the optimizer should be matched not only to the scale of training, but also to the symmetry and geometry of the module it acts upon.

In this sense, recent progress in optimizer design may be interpreted as part of a larger transition from generic coordinate-wise methods toward *module-aware* and *geometry-aware* optimization. Spectral optimizers provide one canonical instance of this principle for matrix-valued parameters with bi-orthogonal symmetry, while the right-spectral and left-spectral extensions developed here illustrate how the same philosophy can be adapted to embeddings, LM heads, MOE routers, and other specialized modules arising in modern architectures. This suggests a future in which deep learning systems are trained not by a single universal optimizer, but by a coordinated family of symmetry-compatible layerwise optimizers tailored to the structural role of each component.

M.2 Training Stabilization Tricks and Heuristics

As model scale increases, coordinate-wise adaptive optimizers are almost never used in isolation. Instead, they are typically accompanied by a growing suite of stabilization tricks and heuristics designed to control training dynamics. Examples include variants such as STABLEADAMW [146] and the numerous practical stabilization techniques used in large-model training recipes [67]. The prevalence of such heuristics highlights an important point: the default optimizer geometry is often not sufficiently well matched to the true parameter geometry of the model, and substantial additional engineering is needed to compensate.

This suggests a broader perspective on stabilization. Rather than treating training heuristics as unrelated auxiliary fixes, one may view many of them as attempts to partially correct a mismatch between optimizer geometry and model geometry. In that sense, geometry-aware optimizers may reduce the need for some of these heuristics by encoding more appropriate invariance and scaling structure directly into the update rule.

M.3 Computational Challenges

The emergence of MUON can be attributed in large part to engineering progress in making numerical polar decomposition, or matrix orthogonalization, computationally practical at scale, especially through the use of the Newton–Schulz iteration. More generally, the future of geometry-aware optimizers depends crucially on advances in numerical linear algebra for matrix decompositions and related primitives. In particular, practical success will favor algorithms that are GPU-friendly, communication-efficient, and based primarily on highly optimized operations such as general matrix multiplications (GEMMs).

Recent progress in this direction includes convergence analyses of improved inexact polar oracles based on Newton–Schulz [75], QDWH [83], and POLAR EXPRESS [5], newer GPU-friendly numerical polar decomposition algorithms such as CANS [56], PRISM [157], Turbo-MUON [16], Flash-MUON [96], and Gram Newton–Schulz [164], as well as quantization of MUON optimizer states [58]. These developments suggest that the computational barrier to matrix-aware optimizers is steadily falling, and that future optimizer research will likely be tightly intertwined with the development of fast matrix decomposition routines and inexact structured oracles.

M.4 Scalability Challenges

Beyond single-device efficiency, the long-term success of geometry-aware optimizers also depends on their amenability to distributed training and to different parallelism strategies. Moving beyond elementwise operations raises new systems challenges: communication patterns, synchronization costs, memory sharding, and compatibility with tensor, pipeline, and data parallelism all become central design considerations. Recent distributed implementations such as Distributed MUON [42, 98, 116], DION [2, 3], DISCO [44], and Parallel MUON [95] show that such challenges can be addressed in practice, while modern large-scale frameworks such as TORCHTITAN [94] with PyTorch FSDP2 [172] and Megatron-LM [133] with Megatron Core [154] provide increasingly realistic substrates for integrating non-elementwise optimizers.

Encouragingly, MUON has already shown promising results in industry-scale large-model pre-training, including those by Moonshot AI [76, 98], Essential AI [43], Prime Intellect [116], Zhipu AI [49, 50], Zyphra [8], Motif Technologies [95], Arcee AI [135], StepFun [136] and DeepSeek-AI [30]. These developments suggest that the question is no longer whether matrix-aware optimizers can be scaled, but rather how far such optimizers can be pushed once algorithmic geometry, numerical linear algebra, and distributed systems are designed in concert.

N Limitations

Our work proposes an equivariance principle for designing optimizers for matrix-valued neural network parameters, but several limitations remain.

First, our theoretical framework focuses primarily on equivariance properties of optimizer update maps rather than on a full characterization of the training objective. In particular, we do not assume that the loss function of a neural network is globally invariant under arbitrary orthogonal transformations of an individual layer parameter. Instead, our analysis asks whether the optimizer update transforms consistently when a matrix gradient or momentum is represented in a rotated or permuted basis. This distinction is important: equivariance of the update map is a weaker and more practically relevant requirement than invariance of the full training objective. Nevertheless, a more complete theory connecting layerwise optimizer equivariance to the global symmetries of deep networks remains an important direction for future work.

Second, although our proposed spectral and one-sided spectral optimizers are motivated by matrix geometry, their practical implementation requires numerical linear algebra routines such as

Newton–Schulz iteration, POLAR EXPRESS, and Gram Newton–Schulz. These routines introduce additional computational overhead and can be sensitive to numerical precision, especially for very large vocabulary-indexed matrices. Our experiments show that row-norm-based optimizers can provide a computationally cheaper symmetry-compatible alternative, but the best tradeoff between spectral accuracy, wall-clock time, and final validation loss is still architecture- and hardware-dependent.

Third, our empirical study is focused on language model pre-training, with particular emphasis on dense and sparse MOE architectures containing large embedding, LM head, and router matrices. While this setting is central to modern large-scale training, it does not exhaust the range of neural network architectures where symmetry-compatible optimizer design may be relevant. For example, convolutional networks, vision transformers, multimodal models, diffusion models, as well as parameter-efficient fine-tuning and post-training objectives may require additional layer-specific symmetry analyses. Extending the proposed framework to these settings is a natural next step.

Fourth, our experiments are necessarily limited in scale relative to frontier training runs. Although we evaluate models ranging from Qwen3-0.6B-style and Gemma 3 1B-style dense models to sparse MOE variants based on OLMOE-1B-7B and downsized gpt-oss, we do not claim to establish a universal scaling law for symmetry-compatible optimizers. The observed gains are consistent across our experiments, but the models differ in architecture, vocabulary size, training length, and data mixture. Larger-scale studies with more extensive hyperparameter sweeps are needed to determine how the benefits scale with parameter count, token budget, vocabulary size, and degree of gradient anisotropy. Our experiments are also based on a single dataset, FineWeb-Edu, despite its high quality.

Fifth, we do not exhaustively tune all optimizer hyperparameters. In particular, spectral and hybrid optimizers introduce additional choices such as the number of inner iterations, the inverse-square-root backend, the momentum parameter, the scaling exponent, and the order in which row-wise and spectral normalizations are composed. Our comparisons are designed to test the main symmetry-compatible optimizer assignments under practical settings, but more extensive tuning could change the relative ordering among row-norm, one-sided spectral, and hybrid variants.

Finally, our framework identifies symmetry compatibility as an important design principle, but it should not be interpreted as the only consideration in optimizer design. Coordinate-wise adaptive methods remain highly effective for many parameter classes, especially vectors, biases, normalization parameters, and small matrices where the relevant matrix geometry may be less dominant. Practical optimizers for large neural networks will likely remain layerwise hybrids that combine coordinate-wise, row-wise, spectral, and possibly curvature-aware updates according to the geometry, size, and computational constraints of each parameter block.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We ensure that the main claims made in the abstract and introduction are coherent with the paper's overall contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed throughout the paper, as well as Section N.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Assumptions and theorems are given in the main text. All proofs are given in Section I.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Open source code implementation will be made available upon the acceptance of the paper. All experimental setting and hyperparameters are given in Section L.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Open access to code will be made available upon the acceptance of the paper. Open data available at HuggingFace are used.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Experimental details are given in Section L.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: No exact error bars are reported. Due to the high cost of pre-training runs of large models for a large number of tokens, we did not run the experiments for more than one seed. We however reckon that the performance gaps between different methods are large enough to be statistical significant.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All experimental details are given in Section L.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors make sure to conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This paper is largely theoretical and foundational research.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The authors have tried their best to cite all related papers and existing assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Open source code implementation will be made available upon the acceptance of the paper. Currently, the implementation of the proposed algorithms are described in detail. The dataset and code used in this paper can all be found online through their open-source implementation, e.g., HuggingFace.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.