MUON IS THEORETICALLY WRONG FOR CONVOLUTIONS, BUT EMPIRICALLY EFFECTIVE

Anonymous authors
Paper under double-blind review

ABSTRACT

Muon, an optimizer known for its efficiency, has a clear interpretation for matrix-valued updates, but convolutional kernels are stored as four-dimensional tensors. Standard implementations reshape these tensors into matrices, a shortcut which breaks the theoretical understanding behind Muon. To investigate this, we formalize the corresponding optimization objective directly in convolutional operator geometry and introduce Convolutional Newton-Schulz (Conv-NS), which approximates the polar factor in this geometry while preserving kernel support. When applied in fast training experiments, Conv-NS and reshape-based Muon are both computationally efficient and achieve comparable accuracy on CIFAR-10 and ImageNet classification tasks. However, as one could expect a theoretically aligned Conv-NS to outperform reshape-based Muon, we investigate this mismatch between practice and theoretical understanding, with the hypothesis that exact convolutional orthogonalization may overconstrain updates. These findings highlight Muon's strong practical performance while opening directions for its further development on convolutions.

1 INTRODUCTION

A recent line of work in deep learning optimization departs from element-wise update rules such as SGD and AdamW and instead exploits the matrix structure of neural network parameters. This perspective has led to influential methods ranging from preconditioned second-order optimizers such as Shampoo (Gupta et al., 2018; Anil et al., 2020), which accumulate Kronecker-factored curvature estimates, to first-order methods such as Muon (Jordan et al., 2024b; Bernstein, 2025). Muon replaces the gradient of a linear layer by an approximation of its semi-orthogonal polar factor, effectively promoting well-conditioned, near-orthogonal updates at the operator level. This optimizer has demonstrated competitive performance across tasks such as transformer pretraining (Jordan et al., 2024a; Karpathy, 2025), image diffusion model training (Schaipp, 2025), and large language model training (AI et al., 2025; Jordan et al., 2024a; Liu et al., 2025). We provide a more detailed discussion of related work in Section A.

While this construction is natural for linear layers, convolutional layers do not fit trivially into the same framework. In this paper, we study the use of Muon on 2D convolutions. A convolutional layer is parameterized by a compact kernel $K \in \mathbb{R}^{c_{\text{out}} \times c_{\text{in}} \times k_h \times k_w}$, but acts on activations through a much larger structured linear operator $\mathcal{T}(K)$, typically Toeplitz or doubly circulant depending on the boundary conditions, that need not be materialized during the forward pass to avoid redundant computations. In practice, current implementations apply Muon by reshaping the 4D kernel into a matrix of size $\mathbb{R}^{c_{\text{out}} \times (c_{\text{in}} k_h k_w)}$ and orthogonalizing this reshaped matrix.

However, orthogonalizing the reshaped kernel does not generally correspond to orthogonalizing the equivalent linear transformation matrix (see fig. 1, upper part). The resulting update is therefore misaligned with the operator geometry of the layer being optimized. We formalize this mismatch in section 2 and introduce the first convolution-aware construction that directly targets the polar factor of the induced operator while preserving the finite support of the kernel. In this paper, we use orthogonalization to refer to procedures that drive the singular values of the convolutional operator toward one (as seen in fig. 1, right).

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

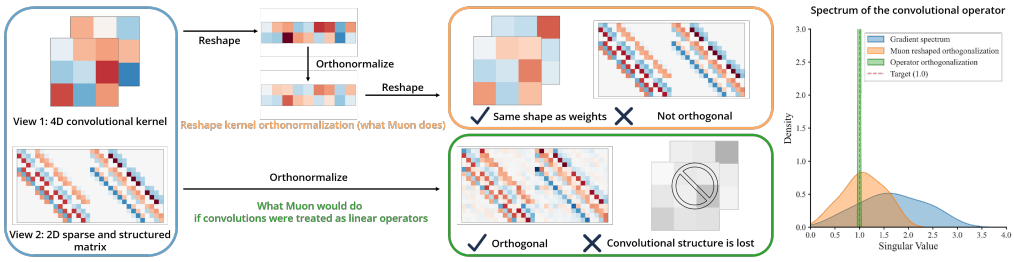

Figure 1: **Problem framing for Muon on convolutions.** Muon's current implementation (option 1) reshape the update and apply PF, however this result in a non orthogonal convolution as observed on the singular value distibution. However there is no trivial solution since the direct application of PF to the equivalent linear operator leads to a an orthogonal matrix, but the structure of the convolution is lost in the process. The 3×3 convolution with $c_i = 2, c_o = 1$ was sampled from a gaussian distribution.

Independently from the less generally understood question of generalization (Wilson et al., 2017; Balles & Hennig, 2018; Luo et al., 2019), we run benchmarks to prioritize speed of convergence (section 5). Surprisingly, reshape-based Muon, despite its structural mismatch, and our convolution-aware alternative Conv-NS achieve comparable performance on these benchmarks, while both substantially outperforming SGD with momentum in compute-constrained settings. These results raise the following question: *why does more faithful convolutional orthogonalization not improve optimization performance?*

The insights we provide are the following:

- **Standard Muon overlooks convolutional geometry.** Applying Muon to convolutions typically involves reshaping each kernel into a matrix before orthogonalization. This does not generally orthogonalize the convolution operator.
- **Convolutional orthogonalization is computationally practical.** We introduce Conv-NS, which preserves kernel support and empirically approaches convolutional orthogonality with near-minimal gradient modification.
- **Better orthogonalization does not consistently improve training.** Conv-NS and reshape-based Muon achieve competitive performance on CIFAR-10 and ImageNet, while increasing Conv-NS iterations can degrade training performance.
- **Extending Muon to convolutions requires more than polar factorization.** We revisit the optimization principle underlying Muon: choosing an update that maximizes predicted loss reduction while limiting its effect on layer outputs. Reshaping does not enforce this limit, while convolutional orthogonality imposes a stronger constraint.
- **These limitations open directions for convolutional Muon.** Both implementations already achieve strong accuracy–runtime trade-offs in the tested settings, motivating updates that better balance predicted loss reduction and control of layer outputs.

2 Muon Foundations and the Convolution Mismatch

Notations For any matrices $X \in \mathbb{R}^{m\times n}$, we denote by $\|X\|_F$ its Frobenius norm, by $\langle X, Y\rangle_F$ the associated inner product, and by $\|X\|_{\text{op}}$ its operator (or spectral) norm. The Stiefel manifold is denoted when $m \leq n$, by $\text{St}(m,n) := \{X \in \mathbb{R}^{m\times n} : XX^\top = I\}$[1], where I denotes the $m \times m$ identity matrix. When the dimensions are clear from the context, we simply write St. We denote the squared Frobenius distance between two matrices $X, Y \in \mathbb{R}^{m\times n}$, by $\mathcal{L}_{\text{frob}}(X, Y) = \|X - Y\|_F^2$.

2.1 About the Muon Update Rule

A dense hidden layer acts on activations via a weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. Given a momentum or gradient matrix G, the Muon optimizer (Jordan et al., 2024b) computes the update direction by approximately orthogonalize the gradient matrix. If $G = U\Sigma V^\top$ is the singular value decomposition, then the Muon update is:

$$\Delta_W^\star = -\eta \,\mathrm{PF}(G) = -\eta UV^\top \tag{1}$$

From an optimization perspective, the polar factor $\mathrm{PF}(G)$ can equivalently be viewed as the projection of G onto the Stiefel manifold St, and thus defined by a constrained minimization over the objective $\mathcal{L}_{\text{frob}}(G, Q)$:

$$\mathrm{PF}(G) = \operatorname*{argmin}_{Q \in \mathrm{St}} \quad \mathcal{L}_{\text{frob}}(G, Q) \tag{2}$$

Since computing an exact SVD at every step is computationally prohibitive, Muon practically approximates $\mathrm{PF}(G)$ using a small number of Newton-Schulz iterations (Björck & Bowie, 1971; Amsel et al., 2025; Grishina et al., 2024)[2]. For an appropriately scaled matrix X, the iteration:

$$X \leftarrow (1+\beta)X - \beta XX^\top X \tag{3}$$

drives the orthogonality penalty $\|XX^\top - I\|_F^2$ iteratively to zero, yielding the polar factor.

2.2 How Standard Muon Handles Convolutions

Convolutional layers do not fit cleanly into this dense picture. The trainable parameters and their gradient updates are finite-support 4D kernels: $G_K \in \mathcal{M}_{\text{kernel}}$, where $\mathcal{M}_{\text{kernel}} = \mathbb{R}^{c_{\text{out}} \times c_{\text{in}} \times k_h \times k_w}$

Unfortunately, materializing the equivalent linear transformation to the convolution is infeasible at scale. Therefore, current implementations of Muon sidestep this by using a structural shortcut: they treat the 4D kernel as a 2D matrix by flattening its spatial dimensions, apply the exact same dense Newton-Schulz orthogonalization routine, and unflatten the result: Denoting the flattening and unflattening operators as $\mathrm{flat} : \mathbb{R}^{c_{\text{out}} \times c_{\text{in}} \times k_h \times k_w} \to \mathbb{R}^{c_{\text{out}} \times (c_{\text{in}} k_h k_w)}$ and $\mathrm{unflat} : \mathbb{R}^{c_{\text{out}} \times (c_{\text{in}} k_h k_w)} \to \mathbb{R}^{c_{\text{out}} \times c_{\text{in}} \times k_h \times k_w}$, the Reshape-Kernel Orthogonalization (RKO) update is:

$$\mathrm{RKO}(G_K) = \mathrm{unflat}\big(\mathrm{PF}(\mathrm{flat}(G_K))\big) \tag{4}$$

This formulation effectively enforces an orthogonality constraint on the flattened kernel matrix. We can formalize the surrogate objective implicitly minimized by RKO as $\mathcal{L}_{\text{flat}}$:

$$\mathcal{L}_{\text{flat}}(\Delta_K) = \|\,\mathrm{flat}(\Delta_K)\,\mathrm{flat}(\Delta_K)^\top - I\|_F^2 \tag{5}$$

By driving $\mathcal{L}_{\text{flat}}$ to zero, RKO makes the flattened matrix perfectly semi-orthogonal. While computationally convenient, $\mathrm{flat}(G_K)$ ignores the true spatial and channel interaction of the convolution, optimizing a mismatched geometric surrogate. For example, it cannot distinguish a 3×3 convolution with $c_{\text{in}} = 8$ and $c_{\text{out}} = 8$ from a pointwise 1×1 convolution with $c_{\text{in}} = 72$ and $c_{\text{out}} = 8$, even though those objects act fundamentally differently on activations. Notably, Wang et al. (2020); Li et al. (2019) show that RKO does not lead to an orthogonal linear operator, as illustrated in the upper part of fig. 1.

3 Our Correction: The Convolutional Polar Factor

To correctly adapt Muon to convolutions, we must work in the space in which the layer actually acts. For a given input activation map, convolution by a kernel K is formally equivalent to a structured linear operator $\mathcal{T}(K) \in \mathbb{R}^{(c_{\text{out}} h_{\text{out}} w_{\text{out}}) \times (c_{\text{in}} h_{\text{in}} w_{\text{in}})}$, typically a doubly circulant or Toeplitz matrix depending on boundary conditions (Achour et al., 2022).

The equivalent Muon update for a convolution layer would therefore be the polar factor (Equation (2)) of the induced operator: $\mathrm{PF}(\mathcal{T}(G_K)) \in \mathbb{R}^{(c_{\text{out}} h_{\text{out}} w_{\text{out}}) \times (c_{\text{in}} h_{\text{in}} w_{\text{in}})}$. As settled for dense matrices, the polar factor of this polar factor $\mathrm{PF}(\mathcal{T}(G_K))$ could be approximated by iterative algorithms

[1] the condition for column orthogonal matrices $n \le m$ is $X^\top X = I$, with I of size $n \times n$

[2] Variants with fifth order matrix multiplication is often used

Algorithm 1 Conv-NS

Require: Gradient kernel G_K, iterations T, step size β

$\Delta_K^{(0)} \leftarrow \mathrm{Normalize}(G_K)$
for $t = 0, \dots, T-1$ **do**
  $C^{(t)} \leftarrow \Delta_K^{(t)} \circledast (\Delta_K^{(t)})^\top \circledast \Delta_K^{(t)}$
  $\widetilde{\Delta}_K^{(t)} \leftarrow (1+\beta)\Delta_K^{(t)} - \beta C^{(t)}$
  $\Delta_K^{(t+1)} \leftarrow \alpha \Pi_{\mathrm{supp}}\left(\widetilde{\Delta}_K^{(t)}\right)$
end for
return $\Delta_K^{(T)}$

Figure 2: **Conv-NS orthogonalizes convolutional updates while preserving kernel support.** Each iteration applies a Newton–Schulz step through block convolutions, then crops the result to the original kernel support.

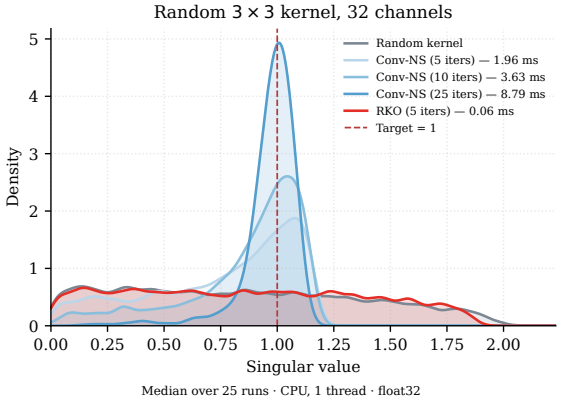

Figure 3: **Conv-NS converges to convolutional orthogonality.** Increasing Conv-NS iterations concentrates the spectrum of singular values around one, whereas RKO, used in standard Muon implementations, leaves it dispersed. All methods start from the same random 3×3 kernel.

(Equation (3)). But, first applying Newton-Schultz on large matrices like $\mathcal{T}(G_K)$ would be costly, and the resulting polar factor generally loses its structural sparsity: $\mathrm{PF}(\mathcal{T}(G_K))$ cannot be converted back into a valid $\mathcal{M}_{\mathrm{kernel}}$ update (lower part of fig. 1). Therefore, a principled definition of the convolutional polar factor requires an extra constraint: the solution operator must remain in the subspace of fixed-support convolutions $\mathcal{M}_{\mathrm{kernel}}$:

$$\mathrm{ConvPF}(G_K) = \underset{\substack{\Delta_K \in \mathcal{M}_{\mathrm{kernel}} \\ \mathcal{T}(\Delta_K) \in \mathrm{St}}}{\mathrm{argmin}} \quad \mathcal{L}_{\mathrm{frob}}(\mathcal{T}(G_K), \mathcal{T}(\Delta_K)). \tag{6}$$

Closeness to the Stiefel manifold is assessed through the semi-orthogonality residual $\|\mathcal{T}(\Delta_K)\mathcal{T}(\Delta_K)^\top - I\|^2$. Wang et al. (2020); Qi et al. (2020); Achour et al. (2022) have shown that this analogously translates to a spatial self-convolution identity on the underlying kernel. We denote this kernel-level orthogonality distance as $\mathcal{L}_{\mathrm{orth}}$:

$$\mathcal{L}_{\mathrm{orth}}(\Delta_K) = \| \mathrm{conv}\big(\Delta_K, \Delta_K, \mathrm{padding} = P, \mathrm{stride} = s\big) - I_{r0}\|_F^2 \tag{7}$$

$$= \frac{1}{h_{\mathrm{out}} w_{\mathrm{out}}} \|\mathcal{T}(\Delta_K)\mathcal{T}(\Delta_K)^\top - I\|_F^2, \tag{8}$$

where I_{r0} is the identity impulse kernel (zero everywhere except at the central position where it is the identity matrix over channels) and $h_{\mathrm{out}} w_{\mathrm{out}}$ is the number of spatial locations. The eq. (8), formally demonstrated in Achour et al. (2022), establishes the exact equivalence[3]. The $\mathrm{ConvPF}(G_K)$ can thus be expressed using the $\mathcal{L}_{\mathrm{orth}}$:

$$\mathrm{ConvPF}(G_K) = \underset{\substack{\Delta_K \in \mathcal{M}_{\mathrm{kernel}} \\ \mathcal{L}_{\mathrm{orth}}(\Delta_K) = 0}}{\mathrm{argmin}} \quad \mathcal{L}_{\mathrm{frob}}(\mathcal{T}(G_K), \mathcal{T}(\Delta_K)). \tag{9}$$

Thus the convolutional polar factor is the equivalent for convolution operators of the target update of Muon for dense matrices. However, neither the SVD decomposition of the operator $\mathcal{T}(G_K)$, nor the RKO heuristic of Muon is able to compute or even approximate $\mathrm{ConvPF}(G_K)$.

4 Convolutional Newton-Schulz: A Convolution-Aware Alternative

We introduce Convolutional Newton-Schulz (Conv-NS), an efficient convolution-aware iterative algorithm that converges to $\mathrm{ConvPF}(G_K)$. Conv-NS serves a dual purpose: it provides a convenient

[3]For simplicity, we assume the row-orthogonal (wide) case $c_{\mathrm{out}} \leq c_{\mathrm{in}} s^2$, analogous to our dense formulation in section 2.1. In the column-orthogonal case, $\mathcal{L}_{\mathrm{orth}}(\Delta_K)$ cannot reach 0 (its global minimum is bounded by $c_{\mathrm{out}} - c_{\mathrm{in}} s^2$), but its minimizer nonetheless yields a semi-orthogonal convolutional operator.

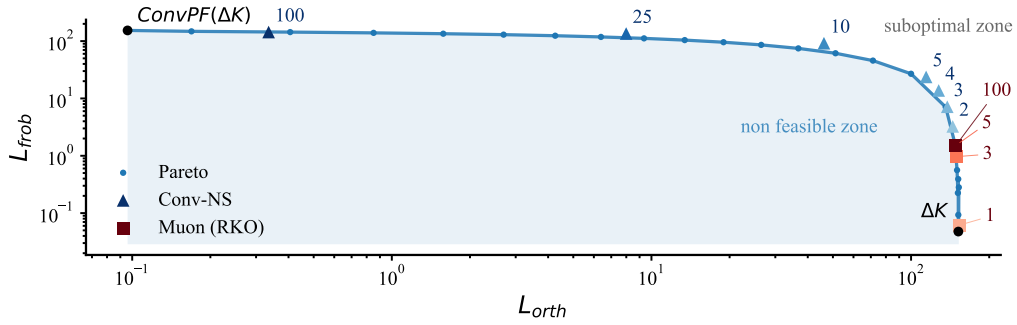

Figure 4: **Conv-NS approaches convolutional orthogonality with near-minimal gradient modification.** Conv-NS follows the numerically estimated orthogonality–fidelity frontier: stronger orthogonality requires progressively larger departures from the original gradient along this frontier. RKO remains near the low-modification end, with higher orthogonality error. Metrics are averaged across convolutional layers and three stages of CIFAR-10 training.

method to orthogonalize the convolution gradient matrix that can be used as a gradient update, evaluated in section 5, and a principled reference for measuring how RKO differs from the true convolutional polar factor. We exploit this second role in section 6 to characterize RKO on real training gradients. The proposed algorithm enables to compute the ground-truth comparison that the existing literature lacks.

We propose an iterative projected gradient descent scheme to jointly drive to orthogonality and constrain to the $\mathcal{M}_{\text{kernel}}$ manifold. After each iteration t of the order-3 Newton-Schulz (also known as Björck) algorithm (Björck & Bowie, 1971), a structural projection Π_{supp} is applied to force the kernel back onto $\mathcal{M}_{\text{kernel}}$:

$$\mathcal{T}(\widetilde{\Delta}_K^{(t)}) \leftarrow (1+\beta)\mathcal{T}(\Delta_K^{(t)}) - \beta\mathcal{T}(\Delta_K^{(t)})\mathcal{T}(\Delta_K^{(t)})^\top\mathcal{T}(\Delta_K^{(t)}) \tag{10}$$

$$\Delta_K^{(t+1)} \leftarrow \alpha\Pi_{\text{supp}}\left(\widetilde{\Delta}_K^{(t)}\right), \tag{11}$$

where β_t is the orthogonalization step size, and α a normalization factor to avoid expansion (justification of this parameter is given in section B). Without the projection (Equation (11)), the kernel size will grow at each iteration, and would converge to $\text{PF}(\mathcal{T}(G_K))$ as described in Section 3.

In practice, these iterations would require huge matrix multiplications with high computational redundancy due to the doubly circulant or Toeplitz structure of $\mathcal{T}(\Delta_K^{(t)})$. We instead propose to use block-convolution to perform the NS-3 scheme directly on the kernels $\Delta_K^{(t)}$. Following Li et al. (2019); Boissin et al. (2025), the *block convolution* operator $\circledast$ combines two kernels into a single kernel representing their composition (without materializing their full convolution matrices). For $A \in \mathbb{R}^{c_{\text{int}}\times c_{\text{in}}\times k_h^A\times k_w^A}$ and $B \in \mathbb{R}^{c_{\text{out}}\times c_{\text{int}}\times k_h^B\times k_w^B}$, their block convolution is defined with respect to the induced linear operators as:

$$\mathcal{T}(B \circledast A) = \mathcal{T}(B)\mathcal{T}(A) \quad \text{with} \quad B \circledast A \in \mathbb{R}^{c_{\text{out}}\times c_{\text{in}}\times(k_h^A+k_h^B-1)\times(k_w^A+k_w^B-1)}. \tag{12}$$

We also use the definition of the transposed convolution kernel $\mathcal{T}(\Delta_K^\top) = \mathcal{T}(\Delta_K)^\top$, which can be computed efficiently in kernel space.

The complete Convolutional Newton-Schultz algorithm is provided as a kernel formulation in Algorithm 1. Lines 3 and 4 are equivalent to the order-3 Newton-Schulz iteration. Importantly, the block convolution expands the spatial support of the operands, so $C^{(t)}$ kernel size is generally larger than $\Delta_K^{(t)}$ one. The projection Π_{supp} crops the expanded kernel by dropping the values outside the original $k_h \times k_w$ support, to converge to $\text{convPF}(\mathcal{T}(G_K))$.

In Section C, we provide the full proofs, together with an explicit gradient formula, showing that the Conv-NS algorithm converges to a minimum of $\mathcal{L}_{\text{orth}}$. section B provide more details about the exact implementation and how strided and grouped convolutions are covered. In Section D, we also propose convolutional adaptations of the main grafting factors commonly applied to Muon updates.

Computing a ground truth for ConvPF. Since ConvPF minimizes gradient modification under convolutional orthogonality (eq. (9)), we construct a numerical reference by relaxing this constraint into a penalty:

$$\min_{\Delta_K \in \mathcal{M}_{\mathrm{kernel}}} \quad \lambda \mathcal{L}_{\mathrm{orth}}(\Delta_K) + (1-\lambda)\mathcal{L}_{\mathrm{frob}}(\Delta_K, G_K), \qquad 0 \le \lambda < 1. \tag{13}$$

Here, G_K is the optimizer momentum rescaled to unit operator norm to separate orthogonalization-induced modification from initial normalization. Varying λ trades fidelity to the original gradient against convolutional orthogonality. Since this objective is nonconvex, we estimate the orthogonality–fidelity frontier using Adam with simulated annealing noise injection for each weighting, retaining the best result across multiple trials (section K.1). This provides a numerical reference for evaluating Conv-NS, rather than a certificate of global optimality.

Empirical validation: Figure 3 compares the singular-value spectra of the RKO Muon update and Conv-NS after different numbers of iterations. It shows that Conv-NS progressively approaches an orthogonal convolution operator. We compare Conv-NS against this numerical reference on synthetic kernels (Figure 8) and real training gradients (Figure 4). For the latter, we collect snapshots at 1%, 50%, and 99% of a CIFAR-10 training run using the architecture of section 5. We evaluate Conv-NS at iteration counts $\{2, 3, 4, 5, 10, 25, 100\}$ against the per-layer frontiers and average metrics across layers and stages. Conv-NS remains close to the estimated frontier, approaching orthogonality with near-minimal gradient modification among the solutions found numerically. Increasing the iteration count moves along this trade-off toward stronger orthogonality and larger gradient modification. Per-stage results are reported in section F.

5 Experiments: CIFAR-10 and ImageNet classification

In the following, we compare Conv-NS Muon against the standard reshape-based Muon on several image-classification benchmarks with convolutional architectures.

Experimental setup. We evaluate Conv-NS in two classification settings that stress different aspects of convolutional optimization. First, CIFAR-10 AirBench94 (Jordan, 2024) is a speed-optimized benchmark tailored to reach 94% accuracy in less than 4 seconds on a single RTX4090. Second, a standard ResNet-50 with a `timm`-style recipe shortened to 30 epochs is trained on ImageNet-1k (Deng et al., 2009). In addition to these standard short-training budgets, we evaluate an extreme 1-epoch setting on both datasets. This 1-epoch regime is specifically designed to isolate early optimization efficiency and highlight differences before network capacity becomes the limiting factor. On CIFAR-10, reshape-based Muon and SGD use the original recipe's heavily optimized settings, while Conv-NS is tuned with a 1000-configuration budget. On ImageNet, reshape-based Muon and Conv-NS receive equal 48-configuration grid-search budgets. Full experimental details are reported in section E; results are summarized in table 1.

Both Muon-Reshape and Muon-ConvNS are efficient. The results highlight that Muon variants drastically outperform SGD in short training settings. This is particularly prevalent in the extreme 1-epoch ImageNet setting, where both Muon (RKO) and the proposed Muon-Conv-NS methods achieve an accuracy above 52%, showcasing a massive gap compared to SGD's 30.47%. This is, to our knowledge, the best result that can be achieved without techniques such as replay buffers (Hu et al., 2021). A deeper comparison with these techniques can be found in section J. However, as training gets longer, performance becomes increasingly capped by the network's capacity rather than the optimizer's speed. This dynamic explains why the performance gap between Muon and SGD narrows in the 8-epoch and 30-epoch settings, and it also contextualizes why SGD remains highly competitive for very long training schedules, such as the 600-epoch runs used in modern ResNet baselines (Wightman et al., 2021), which prioritize generalization.

Conv-NS closes the theory gap with competitive efficiency. Conv-NS successfully fixes the theoretical operator mismatch while achieving performance that is on par with the reshape-based heuristic. On CIFAR-10, Conv-NS matches the accuracy of reshape-based Muon (93.98% vs. 94.02%). On ImageNet, it achieves a competitive 74.53% accuracy, only slightly below reshape-based Muon's 74.82%. In terms of runtime, the principled approach is highly practical: it introduces

Dataset (Budget)	Optimizer	Val. Acc. (%)	Train Acc. (%)	Runtime
CIFAR-10 (8 epochs)	Muon - reshape	94.02 ± 0.14	97.66 ± 0.31	3.074 ± 0.005s
8 layers CNN, $bs = 2000$	Muon - Conv-NS	93.98 ± 0.14	97.17 ± 0.38	3.608 ± 0.018s
1xRTX4090	SGD	91.25 ± 0.09	93.00 ± 0.35	2.975 ± 0.009s
CIFAR-10 (1 epoch)	Muon - reshape	87.18 ± 0.002	86.46 ± 0.01	0.509 ± 0.003s
8 layers CNN, $bs = 250$	Muon - Conv-NS	87.75 ± 0.002	85.44 ± 0.02	0.559 ± 0.003s
1xRTX4090	SGD	84.99 ± 0.002	83.52 ± 0.02	0.496 ± 0.004s
ImageNet (30 epochs)	Muon - reshape	74.82	60.51	1 h 54m 06s ±12m 51s
ResNet-50, $bs = 2048$	Muon - Conv-NS	74.53	58.70	1 h 49m 20s ±1m 42s
4xH100	SGD	70.25	48.03	1 h 33m 56s ±8m 07s
ImageNet (1 epoch)	Muon - reshape	54.05	28.10	14m 57s ±4m 57s
ResNet-50, $bs = 512$	Muon - Conv-NS	52.95	26.74	13m 13s ±44s
1xH100	SGD	30.47	10.10	13m 00s ±3m 46s

Table 1: **CIFAR-10 and ImageNet classification results.** Conv-NS essentially matches reshape-based Muon on CIFAR-10 AirBench94 and remains competitive on ImageNet-1k, while reshape-based Muon keeps a slight overall edge. Train and validation accuracies are shown alongside runtimes for both the full budget (8/30 epochs) and the 1-epoch setting.

only a marginal overhead on the small CIFAR-10 network and actually runs slightly faster than the reshape baseline on the larger ResNet-50 architecture. This demonstrates that closing the theory-practice gap does not require sacrificing computational efficiency. Additional comparisons with other unpublished baselines are reported in section K.2.

6 What should convolutional orthogonalization optimize?

The near-parity with Conv-NS observed on both CIFAR-10 and ImageNet raises a central question: *Why does improving operator orthogonalization in Muon for convolution updates fail to outperform the existing heuristic?*

More accurate orthogonalization can hurt training. We sweep the number of Conv-NS iterations in the one-epoch CIFAR-10 setting of section 5. Under training-loss selection, validation accuracy peaks at 10 iterations (87.56%), then decreases at 25 and 100 iterations (86.42% and 85.13%); training loss also worsens (Table 2). The same qualitative trend holds under validation- and training-accuracy-based selection (Table 6). In the eight-epoch setting, Figure 5 shows that increasing the iteration count substantially reduces $\mathcal{L}_{\mathrm{orth}}$ throughout training, yet the 100-iteration setting generally learns more slowly than intermediate iteration counts. Thus, more accurate convolutional orthogonalization does not consistently improve training performance. Learning-rate sweeps for each iteration count are provided in section H.

Iterations (Conv-NS)	1	5	10	25	100
Val. Acc. (%)	84.93	86.35	**87.56**	86.42	85.13
Train Acc. (%)	86.32	86.72	**87.44**	86.52	86.32
Train Loss	1.1693	1.1519	**1.1352**	1.1571	1.1821
Muon LR	0.579	0.513	0.487	0.200	0.094
Muon Momentum	0.799	0.724	0.250	0.406	0.706

Table 2: **One-epoch CIFAR-10 sweep results.** We report the single top configuration per sweep under training-loss selection; each configuration line is already averaged over repeated runs with the same hyperparameters. The best one-epoch operating point occurs at 10 Conv-NS iterations.

Lower orthogonality errors do not guarantee better training. We train the same CIFAR-10 network for eight epochs with RKO and Conv-NS using $n \in \{1, 5, 10, 25, 100\}$, with five runs per setting. Conv-NS settings share a learning rate of 0.52, momentum 0.6, and damping 0.99;

RKO uses learning rate 0.24 and the same momentum. We track training and validation accuracy alongside $\mathcal{L}_{\text{orth}}$, $\mathcal{L}_{\text{flat}}$, and $\mathcal{L}_{\text{frob}}$, averaged across convolutional layers. RKO's performance raises the question of whether its flattened-kernel objective $\mathcal{L}_{\text{flat}}$ is better suited to training than $\mathcal{L}_{\text{orth}}$. However, Figure 5 shows that high-iteration Conv-NS reduces both objectives and can achieve lower $\mathcal{L}_{\text{flat}}$ than RKO without improving validation accuracy. Thus, lower values of either orthogonality objective alone do not explain the performance ordering. Full residual comparisons are reported in section G. Since Conv-NS empirically approaches convolutional orthogonality with near-minimal gradient modification (Section 4), these results motivate examining whether the ConvPF target itself imposes unnecessary constraints.

Convolutional structure limits the LMO justification for polar factorization. Several works in the literature have proposed explanations for the empirical performance of Muon and gradient orthogonalization. One of the most prominent interpretations relates Muon to approximately solving a Linear Minimization Oracle (LMO) over a spectral-norm ball of radius η[4] (Pethick et al., 2025):

$$\Delta_W^\star = \underset{\|\Delta_W\|_{\text{op}} \le \eta}{\arg\min} \langle G, \Delta_W \rangle_F. \tag{14}$$

This formulation measures the size of the update through the operator norm (spectral norm), thereby constraining the worst-case change in the layer output (Bernstein & Newhouse, 2025). By duality between the spectral and nuclear norms, the closed-form solution is proportional to the semi-orthogonal polar factor of the gradient (Bernstein & Newhouse, 2024; Newhouse, 2025). For completeness, we provide the proof of the optimality of $\text{PF}(G)$ in section L.

For convolutional layers, the LMO must instead be formulated in the operator space in which the layer acts. The minimization is therefore restricted to structured linear operators induced by fixed-support kernels, with $G_K, \Delta_K \in \mathcal{M}_{\text{kernel}}$:

$$\Delta_K^\star = \underset{\substack{\Delta_K \in \mathcal{M}_{\text{kernel}} \\ \|\mathcal{T}(\Delta_K)\|_{\text{op}} \le 1}}{\arg\min} \langle \mathcal{T}(G_K), \mathcal{T}(\Delta_K) \rangle_F. \tag{15}$$

The connection between the convolutional LMO in eq. (15) and the convolutional polar factor $\text{ConvPF}(G_K)$ follows the same proof as for dense matrices (Section L). As a first indication, $\text{ConvPF}(G_K)$ satisfies the unit-ball constraint $\|\mathcal{T}(\Delta)\|_{\text{op}} \le 1$ by construction, unlike the RKO solution (section 2.2). Moreover, decomposing $\mathcal{L}_{\text{frob}}(\mathcal{T}(G_K), \mathcal{T}(\Delta_K))$ shows that $\text{ConvPF}(G_K)$ solves the LMO over the Stiefel-constrained feasible set (Section M):

$$-\text{ConvPF}(G_K) \in \underset{\substack{\Delta_K \in \mathcal{M}_{\text{kernel}} \\ \mathcal{T}(\Delta_K) \in \text{St}}}{\text{argmin}} \langle \mathcal{T}(G_K), \mathcal{T}(\Delta_K) \rangle_F \tag{16}$$

The main difference between Equation (15) and Equation (16) lies in the minimization set which are respectively $\|\mathcal{T}(\Delta)\|_{\text{op}} \le 1$ and $\mathcal{T}(\Delta_K) \in \text{St}$.

Finally, in the dense case, polar-factor optimality follows from $\text{CH}(\text{St})$ (the Convex Hull of St) being the operator-norm unit ball. Does this remain true under the kernel-size constraint in eq. (16)? Indeed, the optimum $\mathcal{T}(\text{ConvPF}(G_K))$ lies both in $\mathcal{T}(\mathcal{M}_{\text{kernel}}) = \{\mathcal{T}(\Delta), \Delta \in \mathcal{M}_{\text{kernel}}\}$, and on the Stiefel manifold St. Therefore, the convex hull of their intersection is contained in the intersection of their convex hulls:

$$\text{CH}(\mathcal{T}(\mathcal{M}_{\text{kernel}}) \cap \text{St}) \subseteq \mathcal{T}(\mathcal{M}_{\text{kernel}}) \cap \text{CH}(\text{St}) = \{\Delta \in \mathcal{M}_{\text{kernel}} \text{ such that } \|\mathcal{T}(\Delta)\|_{op} \le 1\} \tag{17}$$

The right-hand side is exactly the feasible set of the convolutional LMO in eq. (15). Since the inclusion is generally strict, $\text{ConvPF}(G_K)$ *is optimal over a more constrained set than the operator-norm unit ball restricted to* $\mathcal{T}(\mathcal{M}_{\text{kernel}})$.

This distinction offers a possible explanation for the previous results: approaching ConvPF more accurately enforces orthogonality constraints beyond what the convolutional LMO requires. The accompanying increase in gradient modification may therefore hinder early optimization, even as both $\mathcal{L}_{\text{orth}}$ and $\mathcal{L}_{\text{flat}}$ decrease (Table 2 and fig. 5).

[4]In the following, η may be omitted and replaced by the unit ball for simplicity

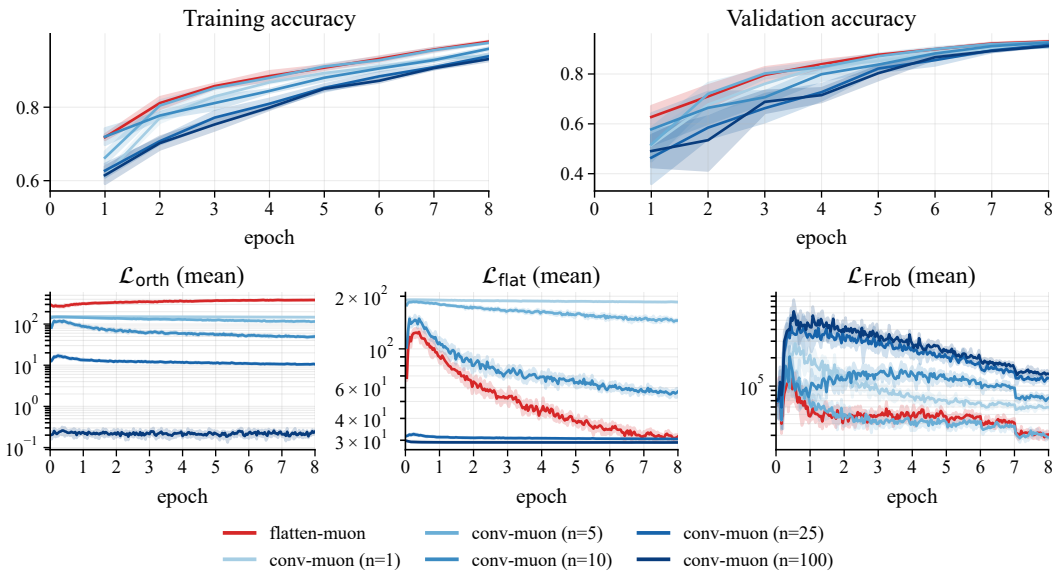

Figure 5: **Better convolutional and flattened orthogonality does not guarantee better training.** Increasing Conv-NS iterations lowers both $\mathcal{L}_{\text{orth}}$ and $\mathcal{L}_{\text{flat}}$, yet high iteration counts slow early learning and increase gradient modification. Curves show means over five eight-epoch CIFAR-10 runs; shaded bands indicate one standard deviation.

Both approaches therefore have limitations: reshape-based Muon does not enforce the convolutional operator-norm constraint (under constraint), while Conv-NS targets the stronger orthogonality constraint (over constraint). Neither is guaranteed to solve the convolutional LMO; a principled extension of Muon to convolutions requires more than computing the convolutional polar factor.

7 Conclusion

In this work, we revisited Muon for convolutional layers from the perspective of the induced convolution operator. We showed that the standard reshape-based implementation does not, in general, orthogonalize this operator.

We introduced Convolutional Newton-Schulz (Conv-NS), a convolution-aware procedure that preserves kernel support and converges toward the convolutional polar factor. Despite enforcing better operator-level orthogonality, Conv-NS only achieves performance comparable to reshape-based Muon on CIFAR-10 and ImageNet. Moreover, increasing the number of Conv-NS iterations could degrade performance, suggesting that exact convolutional orthogonality may overconstrain the update.

We finally relate this observation to the LMO interpretation of Muon: the convolutional polar factor is optimal under the Stiefel constraint, but not necessarily the minimizer of the full operator-norm LMO. This suggests that adapting Muon to convolutions requires balancing operator-level control with sufficient freedom in the update.

Finally, future work could explore more various training tasks like generative modeling or self-supervised representation learning with Conv-NS; along with its extensions for non-2D convolutional layers. Indeed, real training gradient analysis on classification tasks seems to indicate that reshape-based Muon empirically finds a compromise between gradient fidelity and orthogonality, this is not the case on synthetic gradients of different distributions (see fig. 8). This distribution-dependent optimality indicates that reshape-based Muon could be suboptimal on different training tasks with different gradient distributions.

AI USE STATEMENT

In this work, we used generative AI tools for coding assistance in this project, we limited the use of this tooling to boilerplate functions and utilities. Every function or class written with AI assistance was rigorously checked and tested. The main optimizer class is nearly entirely human written except for checked boilerplate code. Additionally, AI tools were used for polishing grammatical errors in the writing of the main paper. AI tooling was used to check the soundness of the existing human written proofs that were checked previously by all authors. AI tooling was also used to improve the visual formatting of pre-existing human written plotting scripts, these scripts *only* changed the formatting of said plots. All concepts and novel ideas in this paper are solely the authors' work. No AI system was used for ideation or concept generation. AI systems were tasked to identify potential gaps in the authors' manually conducted related-work survey. The additional works it found were all manually checked for existence and relevance to the paper's message and studies. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Along with this work, we provide access to the code used to run the experiments of the paper. More specific details are given in section E.

REFERENCES

El Mehdi Achour, François Malgouyres, and Franck Mamalet. Existence, stability and scalability of orthogonal convolutional neural networks. *Journal of Machine Learning Research*, 23(347), 2022.

Essential AI, Ishaan Shah, Anthony M. Polloreno, Karl Stratos, Philip Monk, Adarsh Chaluvaraju, Andrew Hojel, Andrew Ma, Anil Thomas, Ashish Tanwer, Darsh J. Shah, Khoi Nguyen, Kurt Smith, Michael Callahan, Michael Pust, Mohit Parmar, Peter Rushton, Platon Mazarakis, Ritvik Kapila, Saurabh Srivastava, Somanshu Singla, Tim Romanski, Yash Vanjani, and Ashish Vaswani. Practical Efficiency of Muon for Pretraining, May 2025.

Noah Amsel, David Persson, Christopher Musco, and Robert M. Gower. The Polar Express: Optimal Matrix Sign Methods and their Application to the Muon Algorithm. October 2025.

Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. Scalable second order optimization for deep learning. *arXiv preprint arXiv:2002.09018*, 2020.

Lukas Balles and Philipp Hennig. Dissecting Adam: The Sign, Magnitude and Variance of Stochastic Gradients. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 404–413. PMLR, July 2018. ISSN: 2640-3498.

Jeremy Bernstein. Deriving muon, 2025. URL https://jeremybernste.in/writing/deriving-muon.

Jeremy Bernstein and Laker Newhouse. Old Optimizer, New Norm: An Anthology. December 2024. URL https://openreview.net/forum?id=ux18f5nOpD.

Jeremy Bernstein and Laker Newhouse. Modular Duality in Deep Learning. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 3920–3930. PMLR, October 2025. URL https://proceedings.mlr.press/v267/bernstein25a.html. ISSN: 2640-3498.

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

Thibaut Boissin, Franck Mamalet, Thomas Fel, Agustin Martin Picard, Thomas Massena, and Mathieu Serrurier. An adaptive orthogonal convolution scheme for efficient and flexible cnn architectures. In *International Conference on Machine Learning*, 2025.

Jérôme Bolte, Aris Daniilidis, Olivier Ley, and Laurent Mazet. Characterizations of lojasiewicz inequalities and applications. *Transactions of the American Mathematical Society*, 362(6):3319–3363, 2010.

Ali Ebrahimpour Boroojeny, Matus Telgarsky, and Hari Sundaram. Spectrum extraction and clipping for implicitly linear layers. In *International Conference on Artificial Intelligence and Statistics*, pp. 2971–2979. PMLR, 2024.

Franz Louis Cesista. Frequency domain muon for convolutional neural networks: Simplified, 2026. URL https://leloykun.github.io/ponder/freqmuon/.

Blaise Delattre, Quentin Barthélemy, Alexandre Araujo, and Alexandre Allauzen. Efficient bound of lipschitz constant for convolutional layers by gram iteration. In *International Conference on Machine Learning*, pp. 7513–7532. PMLR, 2023.

Blaise Delattre, Quentin Barthélemy, and Alexandre Allauzen. Spectral norm of convolutional layers with circular and zero paddings. *arXiv preprint arXiv:2402.00240*, 2024.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009.

Sara Dragutinović and Rajesh Ranganath. To Use or not to Use Muon: How Simplicity Bias in Optimizers Matters. March 2026.

Kevin Frans, Pieter Abbeel, and Sergey Levine. What really matters in matrix-whitening optimizers? In *OPT 2025: Optimization for Machine Learning*, 2025.

Ekaterina Grishina, Mikhail Gorbunov, and Maxim Rakhuba. Tight and efficient upper bound on spectral norm of convolutional layers. In *European Conference on Computer Vision*, pp. 19–34. Springer, 2024.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pp. 1842–1850. PMLR, 2018.

Huiyi Hu, Ang Li, Daniele Calandriello, and Dilan Gorur. One pass imagenet. In *NeurIPS 2021 Workshop on ImageNet: Past, Present, and Future*, 2021.

Keller Jordan. 94% on CIFAR-10 in 3.29 seconds on a single GPU, 2024. URL http://arxiv.org/abs/2404.00498.

Keller Jordan, Jeremy Bernstein, Brendan Rappazzo, @fernbear.bsky.social, Boza Vlado, You Jiacheng, Franz Cesista, Braden Koszarsky, and @Grad62304977. modded-nanogpt: Speedrunning the nanogpt baseline, 2024a. URL https://github.com/KellerJordan/modded-nanogpt.

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

Michel Journée, Yurii Nesterov, Peter Richtárik, and Rodolphe Sepulchre. Generalized power method for sparse principal component analysis. *Journal of Machine Learning Research*, 11(2), 2010.

Andrej Karpathy. nanochat: The best chatgpt that $100 can buy, 2025. URL https://github.com/karpathy/nanochat.

Ji-Ha Kim. Frequency-domain muon for conv filters - orthogonalizing the operator, 2026. URL https://jiha-kim.github.io/posts/frequency-domain-muon-for-conv-filters/.

Qiyang Li, Saminul Haque, Cem Anil, James Lucas, Roger B Grosse, and Jörn-Henrik Jacobsen. Preventing gradient attenuation in lipschitz constrained convolutional networks. *Advances in neural information processing systems*, 32, 2019.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is Scalable for LLM Training, February 2025. URL http://arxiv.org/abs/2502.16982. arXiv:2502.16982 [cs].

Liangchen Luo, Yuanhao Xiong, and Yan Liu. Adaptive gradient methods with dynamic bound of learning rate. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg3g2R9FX.

Laker Newhouse. Understanding muon, 2025. URL https://lakernewhouse.com/muon.

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training Deep Learning Models with Norm-Constrained LMOs. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 49069–49104. PMLR, October 2025. ISSN: 2640-3498.

Omead Pooladzandi. Orthogonalizing all layers for muon optimizer via psgd's matrixizer, 2026. URL https://github.com/opooladz/matrixizer-cifar10/tree/master.

Haozhi Qi, Chong You, Xiaolong Wang, Yi Ma, and Jitendra Malik. Deep isometric learning for visual recognition. In *International conference on machine learning*, pp. 7824–7835. PMLR, 2020.

Artem Riabinin, Egor Shulgin, Kaja Gruntkowska, and Peter Richtárik. Gluon: Making Muon & Scion Great Again! (Bridging Theory and Practice of LMO-based Optimizers for LLMs). June 2025.

Fabian Schaipp. Optimization benchmark for diffusion models on dynamical systems. In *EurIPS 2025 Workshop on Principles of Generative Modeling (PriGM)*, 2025. URL https://openreview.net/forum?id=demP9pzBDX.

Hanie Sedghi, Vineet Gupta, and Philip M. Long. The singular values of convolutional layers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJevYoA9Fm.

Alexandra Senderovich, Ekaterina Bulatova, Anton Obukhov, and Maxim Rakhuba. Towards Practical Control of Singular Values of Convolutional Layers. *Advances in Neural Information Processing Systems*, 35:10918–10930, December 2022.

Egor Shulgin, Sultan AlRashed, Francesco Orabona, and Peter Richtárik. Beyond the ideal: Analyzing the inexact muon update. In *The Third Conference on Parsimony and Learning (Recent Spotlight Track)*, 2026. URL https://openreview.net/forum?id=veVNVUSieC.

Bhavya Vasudeva, Puneesh Deora, Yize Zhao, Vatsal Sharan, and Christos Thrampoulidis. How muon's spectral design benefits generalization: A study on imbalanced data. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 142276–142311, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/e6231c5f46598cfd09ff1970524e0436-Paper-Conference.pdf.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. Soap: Improving and stabilizing shampoo using adam for language modeling. In *International Conference on Learning Representations*, volume 2025, pp. 93423–93444, 2025.

Jiayun Wang, Yubei Chen, Rudrasis Chakraborty, and Stella X Yu. Orthogonal convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11505–11515, 2020.

Kaiyue Wen, David Hall, Tengyu Ma, and Percy Liang. Fantastic Pretraining Optimizers and Where to Find Them. In *International Conference on Learning Representations*, 2026.

Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. In *NeurIPS 2021 Workshop on ImageNet: Past, Present, and Future*, 2021.

Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The Marginal Value of Adaptive Gradient Methods in Machine Learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Tian Xie, Haoming Luo, Haoyu Tang, Yiwen Hu, Jason Klein Liu, Qingnan Ren, Yang Wang, Wayne Xin Zhao, Rui Yan, Bing Su, Chong Luo, and Baining Guo. Controlled LLM Training on Spectral Sphere, January 2026. URL http://arxiv.org/abs/2601.08393. arXiv:2601.08393 [cs].

A Related Work

A.1 Muon and Matrix-Whitening Optimization

The development of Linear Minimization Oracle (LMO) and matrix-whitening optimizers, such as Muon (Jordan et al., 2024b; Bernstein, 2025) and SOAP (Vyas et al., 2025; Wen et al., 2026), aims to improve upon standard element-wise algorithms like AdamW. Frans et al. (2025) argue that the gains of matrix-whitening optimizers are not explained solely by spectral normalization: in their Transformer language-modeling experiments, variance-adapted variants such as Adam, SOAP, and AdaMuon outperform sign-descent counterparts such as Signum, SPlus, and Muon. Applying such optimizers alters the implicit bias of the network, fundamentally shifting convergence toward max-margin solutions with respect to the spectral norm rather than prioritizing simple dominant components (Dragutinović & Ranganath, 2026; Vasudeva et al., 2026). While this structural bias often benefits generalization on complex distributions, controlling the resulting weight drift remains challenging, leading to variants like Spectral Sphere Optimizer (SSO) (Xie et al., 2026) which enforces strict module-wise spectral properties. Despite reports of substantial compute-time frontier breakthroughs and improved large-batch efficiency (AI et al., 2025), recent rigorous scaling laws show that the speedup relative to AdamW diminishes at larger parameter scales depending on careful hyperparameter tuning (Wen et al., 2026). Furthermore, theoretical stepsizes are often bridged to empirical reality by accounting for inexact matrix updates (Shulgin et al., 2026) and layer-wise geometry (Riabinin et al., 2025).

A.2 Community Explorations for Convolutional Orthogonalization

The structural gap between 4D kernels and 2D spectral optimizers has spurred recent community efforts to adapt Muon for CNNs, notably Freq-Muon (Kim, 2026; Cesista, 2026), which orthogonalizes per-frequency channel-mixing matrices via the DFT, and Matrixizer (Pooladzandi, 2026), which searches for an optimal 2D factorization of the gradient tensor before applying Newton–Schulz. These implementations highlight an active demand for a CNN-compatible Muon, yet no peer-reviewed work has formalized these approaches or provided rigorous evaluations. We describe both methods in detail and report appendix-only comparisons in section K.

B Implementation of Conv-NS

Algorithm 2 Convolutional Newton–Schulz (Conv-NS)

Require: Kernel K, stride $s = (s_h, s_w)$, groups g
Require: $N = 12$, $\beta = 0.5$, $\alpha = 0.91$, $\varepsilon = 10^{-2}$
$(c_o, c_i/g, k_h, k_w) \leftarrow \mathrm{shape}(K)$
$K \leftarrow \textsc{SpectralRescale}(K)$ ▷ Three Gram iterations
$p \leftarrow \big(s_h\lfloor (k_h-1)/s_h \rfloor,\ s_w\lfloor (k_w-1)/s_w \rfloor\big)$
for $t = 1, \ldots, N$ **do**
  $B \leftarrow \textsc{PackGroups}(K, g)$ ▷ Only for grouped convolutions
  $A \leftarrow \textsc{Conv2D}(B, K;\ s, p, g)$
  $C \leftarrow \textsc{ConvTranspose2D}(A, K;\ s, p, g, 0)$ ▷ Performs Convolution and cropping
  $C \leftarrow \textsc{UnpackGroups}(C, g)$ ▷ Only for grouped convolutions
  $K \leftarrow \alpha\big((1+\beta)K - \beta C\big)$
end for
return K/α

We initialize the kernel by spectral rescaling using three Gram iterations (Delattre et al., 2023; 2024). Group packing rearranges the kernel so that grouped convolutions can process its output filters as a batch; unpacking restores the original layout. The convolution–transposed-convolution pair evaluates the cubic Newton–Schulz correction on the original kernel support, with cropping implemented implicitly by the padding choices. We use a constant $\beta = 0.5$: increasing β can destabilize the iteration, whereas decreasing it slows convergence. Although the unprojected polynomial preserves the operator-norm unit ball for $0 < \beta \leq 0.5$, support cropping need not do so. Damping by $\alpha = 0.91$

provides a stability margin intended to keep the iterates inside this ball. The final division by α removes the last damping factor. The block-convolution derivation concerns ungrouped, unit-stride convolutions; the grouped and strided extensions were checked by unit tests.

Efficiency. Cropping preserves the original kernel support, giving each iteration a fixed computational and memory cost for fixed kernel dimensions, stride, and group count. The Conv-NS loop operates directly on real-valued kernels and requires no FFT. It therefore avoids complex-valued frequency-domain arithmetic and transforms on the activation-sized grid, whose spatial dimensions are typically much larger than the kernel support.

C CONVERGENCE ANALYSIS

The convolutional Newton–Schulz update should be interpreted in two stages. Without the support projection, it is exactly the classical Björck / Newton–Schulz iteration written in kernel coordinates. Once the projection onto fixed spatial support is inserted, the method is no longer exact Newton–Schulz on the full convolution operator; it becomes a projected gradient method for a support-constrained orthogonality objective. This objective coincides with the $\mathcal{L}_{\mathrm{orth}}$ surrogate of Achour et al. (2022) in the stride-aware setting.

Exact block-convolution form of Newton–Schulz when $S = 1$. Assume circular padding and unit stride. Let

$$\widehat{K}^{(t+1)} = (1+\beta)\widehat{K}^{(t)} - \beta\, \widehat{K}^{(t)} \circledast (\widehat{K}^{(t)})^{\top} \circledast \widehat{K}^{(t)},$$

where the support of $\widehat{K}^{(t)}$ is allowed to grow with t. Define

$$X^{(t)} := \mathcal{T}_{\mathrm{circ}}(\widehat{K}^{(t)}).$$

Proposition C.1 (Unprojected block-convolution Newton–Schulz)**.** *For every t,*

$$X^{(t+1)} = (1+\beta)X^{(t)} - \beta X^{(t)}(X^{(t)})^{\top}X^{(t)}.$$

Hence the unprojected kernel iteration is exactly the classical Björck / Newton–Schulz iteration on the induced convolution operator. In particular, if $\|X^{(0)}\|_2 \le 1$ and $0 < \beta \le \frac{1}{2}$, then $X^{(t)}$ converges to the polar factor $\mathrm{PF}(X^{(0)})$.

Proof. Using the block-convolution identities,

$$\mathcal{T}_{\mathrm{circ}}(A \circledast B) = \mathcal{T}_{\mathrm{circ}}(A)\mathcal{T}_{\mathrm{circ}}(B), \qquad \mathcal{T}_{\mathrm{circ}}(K^{\top}) = \mathcal{T}_{\mathrm{circ}}(K)^{\top},$$

we obtain

$$\mathcal{T}_{\mathrm{circ}}\big(\widehat{K}^{(t)} \circledast (\widehat{K}^{(t)})^{\top} \circledast \widehat{K}^{(t)}\big) = X^{(t)}(X^{(t)})^{\top}X^{(t)}.$$

Applying $\mathcal{T}_{\mathrm{circ}}$ to the kernel recursion yields the claimed matrix iteration. The convergence statement is then the standard Björck / Newton–Schulz result. □

Implicit Newton–Schulz as gradient descent on the orthogonality objective. Let $\mathcal{M}_{\mathrm{kernel}}$ denote the linear space of kernels with fixed spatial support $k_h \times k_w$, and let Π_{supp} be the orthogonal projection onto that space (center crop). In the row-orthogonal case, define

$$\Phi_{\mathrm{bc}}(K) := \frac{1}{4}\|K \circledast K^{\top} - I\|_F^2, \qquad K \in \mathcal{M}_{\mathrm{kernel}}.$$

The column-orthogonal case is identical after replacing $K \circledast K^{\top} - I$ by $K^{\top} \circledast K - I$.

Proposition C.2 (Gradient formula)**.** *For every $K \in \mathcal{M}_{\mathrm{kernel}}$,*

$$\nabla\Phi_{\mathrm{bc}}(K) = \Pi_{\mathrm{supp}}\big((K \circledast K^{\top} - I) \circledast K\big) = \Pi_{\mathrm{supp}}\big(K \circledast K^{\top} \circledast K - K\big).$$

Consequently,

$$\Pi_{\mathrm{supp}}\Big((1+\beta)K - \beta K \circledast K^{\top} \circledast K\Big) = K - \beta\nabla\Phi_{\mathrm{bc}}(K).$$

Proof. Let $H \in \mathcal{M}_{\mathrm{kernel}}$. Since

$$D(K \circledast K^\top)[H] = H \circledast K^\top + K \circledast H^\top,$$

we have

$$D\Phi_{\mathrm{bc}}(K)[H] = \frac{1}{2}\left\langle K \circledast K^\top - I,\ H \circledast K^\top + K \circledast H^\top \right\rangle.$$

Now use that the adjoint of convolution by $K^\top$ is convolution by K followed by the center crop onto the support of H:

$$\langle H \circledast K^\top, Z\rangle = \langle H, \Pi_{\mathrm{supp}}(Z \circledast K)\rangle.$$

Since $K \circledast K^\top - I$ is self-transpose, the two terms coincide, hence

$$D\Phi_{\mathrm{bc}}(K)[H] = \left\langle \Pi_{\mathrm{supp}}\big((K \circledast K^\top - I) \circledast K\big), H \right\rangle.$$

This identifies the gradient. The final identity follows because $\Pi_{\mathrm{supp}}(K) = K$. □

Remark C.3 (Why $\mathcal{L}_{\mathrm{orth}}$ is the right strided objective). For stride $S > 1$, exact Newton–Schulz on the actual layer operator must be written for

$$A_S(K) := S_S \mathcal{T}(K),$$

and its cubic term is

$$A_S(K)A_S(K)^\top A_S(K) = S_S \mathcal{T}(K)\mathcal{T}(K)^\top S_S^\top S_S \mathcal{T}(K),$$

which is not the same as $S_S \mathcal{T}(K \circledast K^\top \circledast K)$. Therefore, in the strided case, the correct fixed-support objective is not the plain unstrided block-convolution residual but the stride-aware regularizer $\mathcal{L}_{\mathrm{orth}}$ of (Achour et al., 2022), whose gradient is implemented in practice by a standard convolution followed by a transposed convolution. The formula above is recovered exactly when $S = 1$.

Projected descent on the support-constrained $\mathcal{L}_{\mathrm{orth}}$ objective. In the row-orthogonal case define

$$\Phi_S(K) := \frac{1}{4}\mathcal{L}_{\mathrm{orth}}(K) = \frac{1}{4}\left\|\mathrm{conv}(K, K, \mathrm{padding} = P, \mathrm{stride} = S) - I_{r0}\right\|_F^2,$$

with $P = \lfloor (k-1)/S \rfloor S$. The practical update can then be written abstractly as

$$K^{(t+1)} = K^{(t)} - \beta_t \nabla \Phi_S(K^{(t)}),$$

or equivalently, in implementation form,

$$K^{(t+1)} = \Pi_{\mathrm{supp}}\Big(K^{(t)} - \beta_t G(K^{(t)})\Big),$$

where $G(K^{(t)})$ is obtained by one direct convolution and one transposed convolution. When $S = 1$, this reduces to the block-convolution formula above.

Theorem C.4 (Convergence to a stationary point)**.** *Assume that the iterates remain in a bounded set $\mathcal{D} \subset \mathcal{M}_{\mathrm{kernel}}$. Since Φ_S is a polynomial on the finite-dimensional space $\mathcal{M}_{\mathrm{kernel}}$, its gradient is Lipschitz on $\mathcal{D}$; let L be such a Lipschitz constant. If the step sizes satisfy*

$$0 < \underline{\beta} \le \beta_t \le \overline{\beta} < \frac{2}{L},$$

then

$$\Phi_S(K^{(t+1)}) \le \Phi_S(K^{(t)}) - c\|\nabla \Phi_S(K^{(t)})\|_F^2, \qquad c := \underline{\beta}\Big(1 - \frac{L\overline{\beta}}{2}\Big) > 0.$$

Consequently,

$$\sum_{t=0}^{\infty} \|\nabla \Phi_S(K^{(t)})\|_F^2 < \infty, \qquad \nabla \Phi_S(K^{(t)}) \to 0,$$

and every accumulation point of $(K^{(t)})_{t\ge 0}$ is a critical point of the support-constrained problem

$$\min_{K \in \mathcal{M}_{\mathrm{kernel}}} \Phi_S(K).$$

Moreover, because Φ_S is a polynomial (hence a Kurdyka–Łojasiewicz function), the whole sequence $(K^{(t)})_{t\ge 0}$ converges to a single critical point $K^\star$ of Φ_S.

Proof. By the descent lemma for L-smooth functions,

$$\Phi_S(K - \beta\nabla\Phi_S(K)) \leq \Phi_S(K) - \beta\|\nabla\Phi_S(K)\|_F^2 + \frac{L\beta^2}{2}\|\nabla\Phi_S(K)\|_F^2.$$

Applying this with $K = K^{(t)}$ and $\beta = \beta_t$ gives

$$\Phi_S(K^{(t+1)}) \leq \Phi_S(K^{(t)}) - \beta_t\Big(1 - \frac{L\beta_t}{2}\Big)\|\nabla\Phi_S(K^{(t)})\|_F^2,$$

hence the announced inequality. Summing over t proves that $\sum_t \|\nabla\Phi_S(K^{(t)})\|_F^2 < \infty$, and therefore $\nabla\Phi_S(K^{(t)}) \to 0$. Criticality of accumulation points follows by continuity of $\nabla\Phi_S$. Finally, the KL property of polynomial functions upgrades subsequential convergence to convergence of the whole bounded sequence. □

Corollary C.5 (Operator-level consequence)**.** *Under circular boundary conditions and $SN \geq 2k-1$, the results summarized in the* LORTH *analysis imply that $\mathcal{L}_{\text{orth}}$ controls both the Frobenius and spectral orthogonality errors of the induced convolution operator. Therefore, if $\Phi_S(K^{(t)}) \to 0$, then the operator-level orthogonality residuals vanish. More generally, if the terminal value $\Phi_S(K^{(T)})$ is small, then the spectral residual is small, and the induced convolution is approximately isometric. Thus the projected convolutional Newton–Schulz iteration should be viewed as a support-constrained orthogonality solver: when an orthogonal kernel exists, it can converge to such a solution; otherwise it converges to the best stationary approximation inside the fixed-support kernel class.*

Initial conditions. To guarantee the convergence of the Newton–Schulz iteration, the initial operator norm of the convolution must be strictly bounded, satisfying $\|\mathcal{T}(K)\|_2 \leq 1.0$. Although a simple Frobenius normalization of the kernel tensor is sufficient to achieve this, employing a tighter spectral estimate accelerates convergence. The accurate evaluation of the Lipschitz constant for convolutional layers is an active area of research in its own right: initial approaches leveraged the Fast Fourier Transform (FFT) to compute a lower bound for layers with circular padding (Sedghi et al., 2019). This foundation was subsequently refined by methods offering improved computational efficiency (Senderovich et al., 2022), support for diverse padding strategies (Grishina et al., 2024), and the ability to extract broader regions of the operator spectrum (Boroojeny et al., 2024). In our implementation, we rely on the techniques established by Delattre et al. (2023) and Delattre et al. (2024), which allow us to efficiently compute a certifiable upper bound on the spectral norm across various padding modes.

Finally, while the target implies that $\|\mathcal{T}(K)\|_2 \leq 1.0$, the projection step, coupled with numerical imprecision, can yield an iterate with $\|\mathcal{T}(K^{(t)})\|_2 \geq 1.0$. To avoid this, we add a damping factor; when $\beta = 0.5$, a damping factor of 0.9 yields good results.

Convergence guarantees: Without the support projection, the block-convolution Newton-Schulz iteration is *exactly* the classical Björck / Newton–Schulz iteration on the induced operator $\mathcal{T}(\Delta_K)$, and therefore converges to its polar factor whenever $\|\mathcal{T}(\Delta_K)\|_2 \leq 1$ and $\beta \leq 0.5$. Once the projection Π_{supp} is inserted, the method is no longer exact Newton–Schulz on the full operator; instead, it becomes projected gradient descent on a support-constrained orthogonality objective $\Phi_S = \|\mathcal{T}(\Delta)\mathcal{T}(\Delta)^T - I\|^2$ that coincides with the $\mathcal{L}_{\text{orth}}$ surrogate of Achour et al. (2022) in the strided setting (proposition C.2). Because Φ_S is a polynomial on the finite-dimensional kernel space, its gradient is Lipschitz and it satisfies the Kurdyka–Łojasiewicz property: the iterates are guaranteed to converge to a single critical point of Φ_S, with $\sum_t \|\nabla\Phi_S(\Delta_K^{(t)})\|_F^2 < \infty$ (Bolte et al., 2010). In particular, when an orthogonal kernel exists within the fixed-support class, the method can converge to it; otherwise it converges to the best stationary approximation. The initial condition $\|\mathcal{T}(\Delta_K)\|_2 \leq 1$ is enforced via a spectral-norm upper bound computed efficiently following Delattre et al. (2023; 2024), and a damping factor handles residual numerical drift across iterations.

D CORRECTION OF THE GRAFTING FACTOR:

Typical implementations of Muon multiply the orthogonal update by a scalar factor based on the matrix dimensions. While there is no strict consensus, these grafting factors allow balancing learning between layers and help with learning rate transferability. Following the logic of this paper, these

	Jordan et al. (2024b)	Liu et al. (2025)	Bernstein & Newhouse (2025)
Dense matrix ($n \times m$)	$\sqrt{\max\left(1, \frac{n}{m}\right)}$	$0.2\sqrt{\max(n, m)}$	$\sqrt{\frac{n}{m}}$
Reshaped kernel	$\sqrt{\max\left(1, \frac{c_{\text{out}}}{c_{\text{in}} k_w k_h}\right)}$	$0.2\sqrt{\max\left(c_{\text{out}}, c_{\text{in}} k_w k_h\right)}$	$\frac{1}{k_w k_h}\sqrt{\frac{c_{\text{out}}}{c_{\text{in}}}}$
Adaptation for $\mathcal{T}(\Delta_K)$	$\sqrt{\max\left(1, \frac{c_{\text{out}}}{c_{\text{in}} s^2}\right)}$	$0.2\sqrt{\max\left(c_{\text{in}} i_w i_h, c_{\text{out}} \frac{i_w}{s} \frac{i_h}{s}\right)}$	$\sqrt{\frac{c_{\text{out}}}{c_{\text{in}} s^2}}$

Table 3: Grafting factors for Muon updates. The adaptation for $\mathcal{T}(\Delta_K)$ applies the dense-matrix formula using the dimensions of the induced convolution operator, $n = c_{\text{out}} \frac{i_h}{s} \frac{i_w}{s}$ and $m = c_{\text{in}} i_h i_w$.

factors should be computed according to the dimensions of the induced operator $\mathcal{T}(\Delta_K)$ rather than the dimensions of the reshaped kernel K. Table 3 presents different scaling conventions and their adaptation. Note that Bernstein & Newhouse (2025) uses a different logic than the other two by proposing distinct formulas for dense and convolutional layers, where the dense formula does not rely on the kernel sizes k_w and k_h.

E CIFAR-10 AND IMAGENET CLASSIFICATION DETAILS

CIFAR-10 AirBench94. The CIFAR-10 experiment uses the AirBench94 benchmark (Jordan, 2024), a speed-optimized 8-epoch training recipe designed to reach 94% accuracy in only a few seconds on a single GPU. The network is the compact AirBench CNN with three convolutional groups (64→256→256 channels) built from 3×3 stride-1 convolutions. Runs use batch size 2000, 8 training epochs, 3 whitening epochs, horizontal-flip augmentation, and translation augmentation with padding 2. Conv-NS and reshape-based Muon use the same architecture, training schedule, augmentation pipeline, batch size, and non-convolutional optimizer groups; only the convolutional orthogonalization routine inside Muon differs. The SGD baseline replaces Muon entirely.

For reshape-based Muon and SGD, we use the original AirBench author's benchmark-specific optimized settings. Because no reference Conv-NS setting existed for this benchmark, we tune Conv-NS with a 1000-configuration search budget. The selected Conv-NS configuration uses Muon learning rate 0.52, Muon momentum 0.6, zero Muon weight decay, bias learning rate 0.053, head learning rate 0.67, Adam weight-decay scale 6×10^{-6}, and SGD momentum 0.85 for the remaining parameter groups. For the convolutional orthogonalization itself, we use $\beta_{\text{init}} = 0.25$, $\beta_{\text{end}} = 0.5$, 25 Conv-NS iterations, damping 0.99, $\gamma = 0.01$, and $\epsilon = 0.01$.

Conv-NS, reshape-based Muon, and SGD are each reported over 25 independent runs. We report both ordinary validation accuracy and test-time-augmented validation accuracy. On a single RTX4090, the non-optimized PyTorch implementation of Conv-NS runs in 4.45 seconds per run, compared with 3.68 seconds for reshape-based. Runtime with torch compilation enabled are reported in main paper.

ImageNet-1k ResNet-50. The ImageNet experiment uses ImageNet-1k (Deng et al., 2009) with a ResNet-50 backbone and a `timm`-style training recipe shortened to 30 epochs. Training uses batch size 512 across 4 H100 GPUs (so global batch size is 2048), a linear learning-rate schedule with no warmup, TA-Wide augmentation, mixup (α=0.1), and CutMix (α=1.0). We compare the `flatten` strategy (reshape-based Muon) against the `bjorck` strategy (Conv-NS).

Both strategies receive the same grid-search budget over learning rate $\in \{0.01, 0.02, 0.05, 0.1\}$, momentum $\in \{0.8, 0.9, 0.95\}$, weight decay $\in \{2\times10^{-5}, 10^{-4}\}$, and cautious weight decay $\in$ {true, false}. This gives 48 configurations per strategy and 96 runs in total. Both strategies reach their best validation top-1 accuracy at lr$= 0.02$, momentum$= 0.95$, weight decay$= 2\times10^{-5}$, with cautious weight decay enabled.

F PER-STAGE PARETO ANALYSIS ON REAL TRAINING GRADIENTS

Section 6 reports the orthogonality–fidelity Pareto front averaged over three training stages. Figure 6 shows the same analysis separately for gradient snapshots captured at the beginning, middle, and end of CIFAR-10 training. The key observation—that reshape-based Muon (RKO) lies on or very near the Pareto front for real gradients—holds consistently across all three stages, confirming that this is not an artifact of a particular training phase.

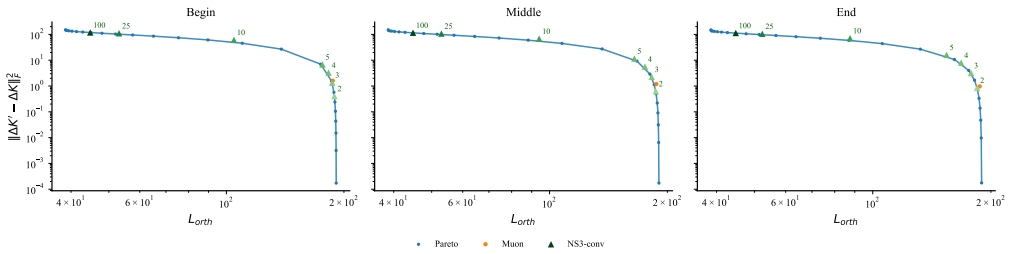

Figure 6: **Per-stage Pareto trade-off on real training gradients.** Each panel shows the Pareto front computed on gradient snapshots from a single training stage (begin, middle, end). Reshape-based Muon (RKO) lies on the Pareto front at all stages, and Conv-NS traces the front as the iteration count increases.

G HIERARCHICAL OPTIMIZATION OF ORTHOGONALITY OBJECTIVES

This section provides the full numerical results comparing the orthogonality residuals of different operators, as discussed in the main text. Table 4 presents the results for synthetic, randomly initialized kernels, while Table 5 shows the results for real gradient snapshots captured during training.

On synthetic kernels, table 4 illustrates an asymmetric relationship between the two orthogonality objectives. RKO nearly eliminates the flattened-kernel residual ($\mathcal{L}_{\text{flat}} = 0.02$), but leaves substantial convolutional error ($\mathcal{L}_{\text{orth}} = 14.40$). Rescaling RKO to unit operator norm reduces this error to 9.13 while sacrificing flattened orthogonality, showing that spectral normalization alone does not resolve the mismatch. In contrast, ConvPF achieves low residuals on both objectives (0.10 and 0.02, respectively), while ten Conv-NS iterations yield intermediate values (3.38 and 1.11). These results show that, in this setting, convolutional orthogonalization can also satisfy the flattened-kernel objective, whereas optimizing the latter alone is insufficient to ensure convolutional orthogonality.

Node	Trials	Lipschitz	L_{orth}	L_{flat}
K	10	2.31 ± 0.07	36.57 ± 1.53	7.11 ± 0.69
$RKO(\Delta K)$	10	1.83 ± 0.03	14.40 ± 0.40	0.02 ± 0.00
$RKO(\Delta K)/\sigma(RKO(\Delta K))$	10	1.00 ± 0.00	9.13 ± 0.14	7.85 ± 0.20
$ConvPF(\Delta K)$	10	1.07 ± 0.02	0.10 ± 0.03	0.02 ± 0.00
$ConvNS_{10}(\Delta K)$	10	1.14 ± 0.01	3.38 ± 0.09	1.11 ± 0.04

Table 4: **Orthogonality hierarchy on synthetic data.** Highlighting the difference between the convolutional polar factor and the reshape-based orthogonalization (RKO). By measuring both the operator-level orthogonality residual $\mathcal{L}_{\text{orth}}$ (Polar factor objective) and the flattened-kernel orthogonality residual $\mathcal{L}_{\text{flat}}$ (RKO objective) we can observe a hierarchy between the two: the convolutional polar factor achieves both operator-level and flattened orthogonality, while RKO only satisfies the latter.

On real training gradients, table 5 confirms the mismatch between flattened and convolutional orthogonality across all three training stages: RKO maintains a low flattened-kernel residual, yet its operator norm remains close to 3, compared with 1.83 in the synthetic setting. ConvPF consistently achieves near-zero convolutional error, while ten Conv-NS iterations provide an intermediate level of orthogonalization. However, the relationship between the two objectives is less clear-cut than on synthetic kernels: ConvPF yields $\mathcal{L}_{\text{flat}} \approx 29$, substantially above RKO's 0.22–0.51. Thus, although

Stage	Node	Trials	Lipschitz	L_{orth}	L_{flat}
Begin	ΔK	6	137.34 ± 47.94	5.2e+07	1.1e+07
Begin	$\Delta K/\sigma(\Delta K)$	6	1.00 ± 0.00	152.45 ± 119.32	191.04 ± 104.36
Begin	$RKO(\Delta K)$	6	2.99 ± 0.04	461.04 ± 304.04	0.51 ± 0.78
Begin	$RKO(\Delta K)/\sigma(RKO(\Delta K))$	6	1.00 ± 0.00	118.74 ± 103.28	151.18 ± 81.79
Begin	$ConvPF(\Delta K)$	6	1.08 ± 0.07	0.09 ± 0.08	29.26 ± 62.44
Begin	$ConvNS_{10}(\Delta K)$	6	1.28 ± 0.09	108.20 ± 101.75	137.14 ± 87.38
Middle	ΔK	6	119.15 ± 47.63	4.7e+07	1.2e+07
Middle	$\Delta K/\sigma(\Delta K)$	6	1.00 ± 0.00	150.68 ± 117.47	189.18 ± 102.69
Middle	$RKO(\Delta K)$	6	2.94 ± 0.12	471.96 ± 287.20	0.22 ± 0.11
Middle	$RKO(\Delta K)/\sigma(RKO(\Delta K))$	6	1.00 ± 0.00	117.73 ± 101.64	149.60 ± 80.70
Middle	$ConvPF(\Delta K)$	6	1.05 ± 0.07	0.04 ± 0.07	29.25 ± 62.47
Middle	$ConvNS_{10}(\Delta K)$	6	1.26 ± 0.10	71.22 ± 61.30	87.57 ± 58.03
End	ΔK	6	126.52 ± 50.66	6.0e+07	1.5e+07
End	$\Delta K/\sigma(\Delta K)$	6	1.00 ± 0.00	150.98 ± 117.89	189.51 ± 103.03
End	$RKO(\Delta K)$	6	2.95 ± 0.11	457.84 ± 281.82	0.22 ± 0.12
End	$RKO(\Delta K)/\sigma(RKO(\Delta K))$	6	1.00 ± 0.00	117.51 ± 101.34	149.55 ± 80.27
End	$ConvPF(\Delta K)$	6	1.04 ± 0.05	0.08 ± 0.07	29.25 ± 62.45
End	$ConvNS_{10}(\Delta K)$	6	1.26 ± 0.10	72.59 ± 64.10	89.26 ± 59.25

Table 5: **Orthogonality hierarchy on real gradients.** The same comparison as in table 4, evaluated on real gradient snapshots from a CIFAR-10 training run across different stages (begin, middle, end). The hierarchy between the objectives remains consistent with the synthetic setting.

the convolutional-orthogonality ordering persists throughout training, these measurements do not establish that ConvPF also minimizes the flattened-kernel residual on real gradients.

H Effect of iteration count in the 8-epoch setting

In Figure 7 the 8-epoch CIFAR-10 AirBench setting is reused to evaluate the effect of different iteration counts. For a given iteration count, the learning rate is swept along the momentum. Five runs are conducted per learning rate, and the momentum with the best average (test time augmentation) validation accuracy across learning rates is selected for each iteration count; shaded bands show ± 1 standard deviation over these five runs. While low iteration budget exhibits a similar behavior observed by (Shulgin et al., 2026): a larger number of iterations improves performance with a shift in optimal lr and momentum, which can be interpreted as the result of a faster training.

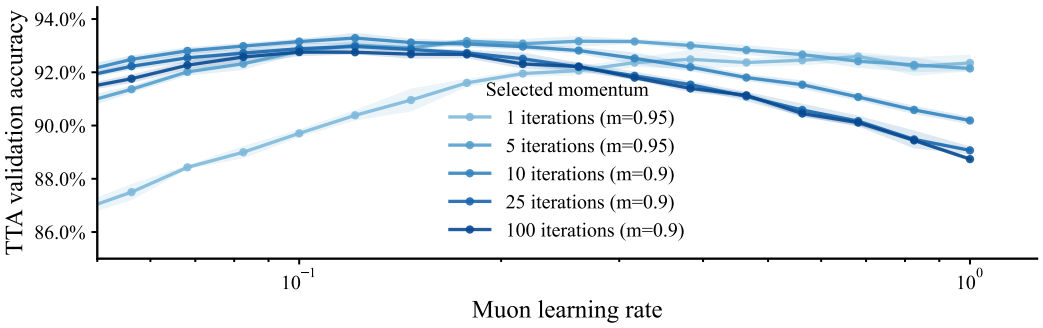

Figure 7: **Effect of Conv-NS iteration count on CIFAR-10 AirBench.** TTA validation accuracy over the Muon learning-rate sweep for different numbers of Newton–Schulz iterations. For each iteration count, momentum is selected by the best average TTA validation accuracy across learning rates; shaded bands show ± 1 standard deviation over five runs.

I ONE-EPOCH SELECTION CRITERIA

Robustness to the selection criterion. To verify that the non-monotonic effect of Conv-NS iteration count is not an artifact of the hyperparameter-selection metric, we reselect the best configurations using either TTA (test time augmentation) validation accuracy or training accuracy (Table 6). In both cases, performance improves up to 10 iterations and degrades at 25 and 100 iterations, despite substantial shifts in the selected learning rate. Moreover, the same trend under training-accuracy selection shows that the degradation is already present at the optimization level rather than arising solely from generalization. These results indicate that the Conv-NS iteration count is not merely an approximation-accuracy parameter: pushing the update closer to exact convolutional orthogonality can itself hinder training.

Top by TTA validation accuracy					
Iterations (Conv-NS)	1	5	10	25	100
TTA Val. Acc. (%)	85.36	86.66	**87.75**	86.61	85.90
Val. Acc. (%)	84.67	85.91	86.84	85.58	84.97
Train Acc. (%)	84.44	86.20	85.44	85.44	85.76
Muon LR	0.433	0.432	0.381	0.181	0.143

Top by training accuracy					
Iterations (Conv-NS)	1	5	10	25	100
TTA Val. Acc. (%)	84.93	86.46	87.55	86.05	85.13
Val. Acc. (%)	84.22	85.71	86.69	85.29	83.94
Train Acc. (%)	86.32	87.32	**88.12**	87.00	86.32
Muon LR	0.579	0.194	0.441	0.158	0.094

Table 6: **Alternate one-epoch CIFAR-10 selection criteria.** These blocks report the single top configuration per sweep under validation-oriented and train-accuracy-oriented selection criteria; each configuration line is already averaged over repeated runs with the same hyperparameters.

J COMPARISON WITH ONE PASS IMAGENET

One Pass ImageNet (OPIN; Hu et al. (2021)) studies ResNet-50 training from a randomly ordered ImageNet stream in which each training example arrives once. Its one-pass constraint permits retaining and revisiting examples in a limited replay buffer. The proposed error-prioritized replay method preferentially samples examples with larger predictive errors and applies importance weights to correct the sampling bias. With k replay minibatches per incoming minibatch, the effective training budget is $E_{\text{eff}} = k + 1$ epochs. OPIN Table 2 reports results at 2, 4, 6, and 9 effective epochs, using buffers containing 1%, 5%, or 10% of the dataset, alongside full-storage multi-epoch baselines.

We compare these results with our one-epoch ImageNet experiments, which use no replay and therefore have $E_{\text{eff}} = 1$. To distinguish data-processing cost from optimization steps, we report both budgets relative to a 90-epoch run with batch size 128. For dataset size N and global batch size B, our batch size of 512 gives approximately 1.11% of the example-processing budget and 0.278% of the update budget. These quantities describe a single training run and exclude hyperparameter search. The computation proxy does not account for optimizer overhead and should not be interpreted as measured FLOPs or wall-clock time.

Both Muon variants exceed all OPIN results at two effective epochs, including the full-storage baseline (46.1%), while processing half as many training examples and taking approximately eight times fewer optimizer updates, without a replay buffer. They remain below the four-effective-epoch replay results (55.5–57.2%). Thus, the comparison highlights strong early learning under a smaller data-processing and update budget. Because the training recipes differ, it is a cross-paper comparison rather than a controlled optimizer ablation; the resource ratios alone do not establish an equivalent wall-clock speedup.

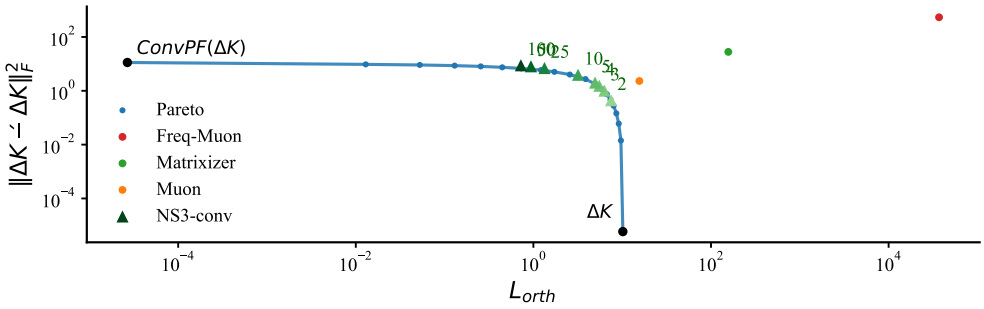

Figure 8: **Pareto trade-off on synthetic Gaussian kernels.** The grey curve is the empirical Pareto front computed via simulated annealing. Conv-NS iterates (blue dots, labelled by iteration count) trace the front from the rescaled SGD corner toward the convolutional polar factor as the number of iterations increases. Reshape-based Muon (RKO) lies far inside the suboptimal zone, reflecting a structural misalignment with the operator-level objective on random kernels. Matrixizer and Freq-Muon are far off-scale.

K Community baseline comparisons

We compare Conv-NS against two recent community-driven baselines -the only publicly available attempts at convolution-aware Muon to date -neither of which has been peer-reviewed or formally evaluated. These comparisons are included for context, but remain appendix material because the main empirical claims center on Conv-NS, reshape-based Muon, and SGD.

Freq-Muon (Kim, 2026; Cesista, 2026). Under circular boundary conditions, a convolution operator is block-circulant and is therefore diagonalized by the spatial DFT into M^2 independent channel-mixing matrices $\hat{K}[\omega] \in \mathbb{C}^{c_{\text{out}} \times c_{\text{in}}}$, one per frequency bin ω. Freq-Muon applies the standard Muon NS5 polar projection independently to each $\hat{K}[\omega]$ and recovers the spatial kernel via inverse FFT. When the FFT grid matches the full input spatial dimensions ($M = i_h = i_w$), this procedure yields an exact orthogonalization of the circular convolution operator. In practice, however, M is set to a small fixed value (e.g. M=8) to keep costs tractable, breaking the exact operator correspondence and turning the method into a surrogate. Additionally, the inverse FFT result is cropped to the original $k_h \times k_w$ support, discarding spatial taps beyond the kernel size.

Matrixizer (Pooladzandi, 2026). Instead of the standard $c_{\text{out}} \times (c_{\text{in}} k_h k_w)$ flattening used in reshape-based Muon, Matrixizer searches over all permutations of the gradient tensor's dimensions

Method	Effective epochs	Computation proxy	Relative updates	Batch size	Updates (approx.)	Validation top-1 accuracy (%) No replay	Replay 1%	Replay 5%	Replay 10%	Multi-epoch 100%
OPIN	2	2/90	2/90	128	20,018	–	44.7	45.1	45.7	46.1
OPIN	4	4/90	4/90	128	40,036	–	55.5	57.1	57.2	59.0
OPIN	6	6/90	6/90	128	60,055	–	58.9	61.3	62.2	64.1
OPIN	9	9/90	9/90	128	90,082	–	59.3	63.2	65.0	68.2
Muon (reshape), ours	1	1/90	1/360	512	2,502	**54.05**	–	–	–	–
Muon (Conv-NS), ours	1	1/90	1/360	512	2,502	52.95	–	–	–	–
SGD, our recipe	1	1/90	1/360	512	2,502	30.47	–	–	–	–

Table 7: Comparison of OPIN Table 2 with the one-epoch ImageNet results in our Table 1. All experiments use ResNet-50. Replay column percentages indicate buffer capacity relative to the dataset; the multi-epoch baseline stores the full dataset. Our runs require no replay storage. Both budget ratios use a 90-epoch, batch-128 reference. Update counts are estimates using $N = 1{,}281{,}167$, rounded to the nearest integer; exact counts depend on final-batch handling. The OPIN counts interpret each reported replay step as a separate minibatch update.

and all possible split points to find the 2D factorization that minimizes $m^2 + n^2$, i.e. the total preconditioner cost (following the PSGD logic of Pooladzandi (2026)). The standard Muon NS5 iteration is then applied to this optimally shaped matrix. While the resulting reshape is typically better balanced than the default flattening, the method remains fundamentally reshape-based: it does not account for stride, groups, dilation, or the structured embedding $\Delta K \mapsto \mathcal{T}(\Delta K)$ that governs how the convolution acts on data.

K.1 Synthetic analysis of the orthogonality–fidelity trade-off

Computing the empirical Pareto front. To evaluate how different orthogonalization strategies navigate the inherent trade-off between operator-level orthogonality ($\mathcal{L}_{\text{orth}}$) and fidelity to the original kernel ($\mathcal{L}_{\text{frob}} = \|\Delta K' - \Delta K\|_F^2$), we first construct a ground-truth Pareto front on synthetic data. We initialize a random Gaussian convolution kernel with $c_{\text{in}} = 16$, $c_{\text{out}} = 16$, and $k_h = k_w = 3$. Because the scalarized objective $\lambda\mathcal{L}_{\text{orth}} + (1-\lambda)\mathcal{L}_{\text{frob}}$ offers no convexity guarantees, we rely on a simulated annealing approach to explore the landscape. We sample 50 scalar weightings λ and optimize each configuration using Adam with simulated annealing noise injection over 1500 iterations. The best result is kept across multiple trials to avoid local minima, yielding an empirical Pareto front that serves as a rigorous baseline.

Evaluating baselines on the synthetic landscape. Against this optimal trade-off curve, we overlay the operating points of Convolutional Newton–Schulz (Conv-NS) at iteration counts $\{2, 3, 4, 5, 10, 25, 50, 100\}$, reshape-based Muon (RKO), Matrixizer (Pooladzandi, 2026), and Freq-Muon (Cesista, 2026). The results are reported in fig. 8.

Conv-NS traces the Pareto front. As the iteration count grows from 2 to 100, the Conv-NS operating points slide along the simulated-annealing Pareto front, progressing from the low-modification region (high $\mathcal{L}_{\text{orth}}$, low $\mathcal{L}_{\text{frob}}$) toward the exact convolutional polar factor (low $\mathcal{L}_{\text{orth}}$, high $\mathcal{L}_{\text{frob}}$). This confirms that the projected iteration efficiently navigates the support-constrained landscape and that the iteration count serves as a user-controllable dial for the orthogonality–fidelity trade-off.

Muon (RKO) is misaligned with the operator-level objective. Unlike its behavior on real training gradients (fig. 4), reshape-based Muon sits far above the Pareto front on synthetic kernels. It modifies the kernel ($\mathcal{L}_{\text{frob}} \approx 2.3$) without improving operator-level orthogonality ($\mathcal{L}_{\text{orth}} \approx 15.6$, compared to $\sim$10.2 for the unmodified kernel). Rather than a systematic degradation, this reflects a structural *misalignment*: on unstructured random Gaussian data, the reshape surrogate pushes the kernel in a direction that does not coincide with the true operator-level objective. A contributing factor is operator-norm inflation—orthogonalizing the flattened matrix can increase the spectral norm of the induced convolution—though the magnitude of this effect varies across initializations. Importantly, as shown in section 6, the structure of real training gradients corrects this misalignment, allowing RKO to fall surprisingly close to the Pareto front in practice.

Matrixizer and Freq-Muon are far from optimal. Matrixizer ($\mathcal{L}_{\text{orth}} \approx 157$) and Freq-Muon ($\mathcal{L}_{\text{orth}} \approx 3.7 \times 10^4$) lie orders of magnitude above the Pareto front, indicating that the approximations underlying these methods—Kronecker factorization and partial-spectrum FFT orthogonalization, respectively—introduce substantial artifacts on this objective.

Practical take-away. Even a modest iteration budget (e.g. 5 iterations) already places Conv-NS near the Pareto front at a useful orthogonality level ($\mathcal{L}_{\text{orth}} \approx 4.9$), offering a practical compromise between compute cost and operator-level fidelity. In contrast, no single-shot baseline achieves a comparable position on the front.

K.2 CIFAR-10 community baseline results

All Muon-family methods: Conv-NS, reshape-based Muon, Matrixizer, and Freq-Muon, share the same architecture, training schedule, and augmentation pipeline; only the orthogonalization routine inside the optimizer differs. Muon - Conv-NS and Muon - reshape are each reported over 25 independent runs (results slightly differs from table 1 which report statistics over 10 runs); Matrixizer,

Method	Val. acc. (%)	TTA acc. (%)	Runs
Conv-NS (ours)	93.17 ± 0.12	93.98 ± 0.09	25
Muon (reshape)	93.10 ± 0.14	93.99 ± 0.14	25
Matrixizer	92.96	93.73	1
Freq-Muon	92.55	93.06	1
SGD	90.47	91.11	1

Table 8: **CIFAR-10 AirBench94 community baseline comparison.** Mean $\pm$ standard deviation over 25 runs for Conv-NS and Muon (reshape); single-run baselines are preliminary and shown as context.

Freq-Muon, and SGD are preliminary single-run results that will be extended before the final version, results are reported form their repositories in table 8.

Conv-NS and reshape-based Muon achieve near-identical CIFAR-10 AirBench94 performance and both outperform the single-run SGD, Freq-Muon, and Matrixizer baselines. Because Matrixizer and Freq-Muon are preliminary community implementations without peer-reviewed standardized evaluations, we use these results as diagnostic context rather than as a central comparison in the main paper.

L LMO DENSE MATRIX PROOF

The objective of Muon is to find for a given layer of weight W the optimal weight update that maximise the loss decrease under a constraint of the maximal variation of the output of the layer. Mathematically, the formulation of the loss decrease is based on the first-order Taylor approximation.

$$\mathcal{L}(W + \Delta_W) = \mathcal{L}(W) + \langle G, \Delta_W \rangle_F + o(\|\Delta_W\|) \,. \tag{18}$$

While the constraint on the maximum variation of the layer output is given by the operator norm:

$$\sup_{\|x\|_2=1} \|Wx - (W + \Delta_W)x\|_2 = \|\Delta_W\|_{\text{op}} \tag{19}$$

This lead to a LMO formulation in the space of the dense matrices:

$$\Delta_W^\star = \underset{\|\Delta_W\|_{\text{op}} \leq \eta}{\arg\min} \langle G, \Delta_W \rangle_F$$

The solution of this LMO is proportional to the the polar factor of the gradient defined by

$$\text{PF}(G) = \underset{Q}{\text{argmin}} \quad \mathcal{L}_{\text{frob}}(G, Q) \quad \text{subject to} \quad \|QQ^\top - I\|_F^2 = 0$$

or with $m \leq n$, using the definition of the Stiefel Manifold $St(m, n) = \left\{Q \in \mathbb{R}^{m \times n} : QQ^T = Id\right\}$, noted St.

$$\text{PF}(G) = \underset{Q \in \text{St}}{\text{argmin}} \quad \mathcal{L}_{\text{frob}}(G, Q)$$

Proof. First, $\forall Q \in \text{St}$, the Frobenius norm is constant and equal to m: $\|Q\|_F^2 = tr(QQ^T) = m$.

Therefore,

$$\|Q - G\|_F^2 = \|Q\|_F^2 + \|G\|_F^2 - 2\langle G, Q \rangle_F = m + \|G\|_F^2 - 2\langle G, Q \rangle_F$$

Thus $PF(G) \in \underset{Q \in \text{St}}{\text{argmax}} \quad \langle G, Q \rangle_F$, or equivalently $-\eta PF(G) \in \underset{Q \in \eta\text{St}}{\text{argmin}} \quad \langle G, Q \rangle_F$.

But in the previous equation, the minimization is over ηSt, and not the set $\{\Delta \in \mathbb{R}^{m \times n} : \|\Delta\|_{\text{op}} \leq \eta\}$.

Since, Linear minimization does not distinguish a set from its convex hull, noted CH, so:

$$-\eta PF(G) \in \underset{Q \in \eta\, \text{CH}(St(m,n))}{\text{argmin}} \langle G, Q \rangle_F$$

.

An important property of the Stiefel Manifold for dense matrices is that:

$$\mathrm{CH}(St(m,n)) = \left\{\Delta \in \mathbb{R}^{m\times n} : \|\Delta\|_{\mathrm{op}} \leq 1\right\}$$

The proof is given in Journée et al. (2010)

Thus

$$-\eta PF(G) \in \underset{\|Q\|_{\mathrm{op}}\leq\eta}{\operatorname{argmin}} \quad \langle G, Q\rangle_F$$

. □

M CONVOLUTIONAL LMO PROOF

This section presents the convolutional LMO construction. Given a convolutional layer with a kernel K, the first-order Taylor approximation is similar to the dense formulatin (Equation (18)):

$$\mathcal{L}(K + \Delta_K) = \mathcal{L}(K) + \langle G_K, \Delta_K\rangle_F + o(\|\Delta_K\|)\,. \tag{20}$$

But the constraint on the minimum variation has to be evaluated in the space of the structured linear operator $\mathcal{T}(K)$:

$$\sup_{\|x\|_2=1} \|\mathcal{T}(K)x - \mathcal{T}(K+\Delta_K)x\|_2 = \|\mathcal{T}(\Delta_K)\|_{\mathrm{op}} \tag{21}$$

The convolutional LMO formulation is thus:

$$\Delta_K^\star = \underset{\substack{\Delta_K\in\mathcal{M}_{\mathrm{kernel}}\\ \|\mathcal{T}(\Delta_K)\|_{\mathrm{op}}\leq\eta}}{\arg\min} \quad \langle G_K, \Delta_K\rangle_F$$

Furthermore, under circular padding and fixed spatial resolution, the Frobenius distance between two convolutional operators is proportional to the Frobenius distance between their underlying kernels:

$$\langle \mathcal{T}(G_K), \mathcal{T}(\Delta_K)\rangle_F \propto \langle G_K, \Delta_K\rangle_F\,. \tag{22}$$

This lead to the convolutional LMO formulation provided in Equation (15).

The link with the convolutional polar factor $\mathrm{ConvPF}(G_K)$ (Equation (6)) follows the same proof as for dense matrices:

Proof. First, $\forall\Delta_K \in \mathcal{M}_{\mathrm{kernel}}$ such that $\mathcal{T}(\Delta_K) \in \mathrm{St}$, the Frobenius norm of $\mathcal{T}(\Delta_K)$ is constant.

Therefore,

$$\|\mathcal{T}(\Delta_K)-\mathcal{T}(G_K)\|_F^2 = \|\mathcal{T}(\Delta_K)\|_F^2+\|\mathcal{T}(G_K)\|_F^2-2\langle\mathcal{T}(G_K),\mathcal{T}(\Delta_K)\rangle_F = cste-2\langle\mathcal{T}(G_K),\mathcal{T}(\Delta_K)\rangle_F$$

Thus $\mathrm{ConvPF}(G_K) \in \underset{\substack{\Delta_K\in\mathcal{M}_{\mathrm{kernel}}\\ \mathcal{T}(\Delta_K)\in\mathrm{St}}}{\operatorname{argmax}} \quad \langle\mathcal{T}(G_K),\mathcal{T}(\Delta_K)\rangle_F \quad ,$

or equivalently $-\eta\,\mathrm{ConvPF}(G) \in \underset{\substack{\Delta_K\in\mathcal{M}_{\mathrm{kernel}}\\ \mathcal{T}(\Delta_K)\in\eta\mathrm{St}}}{\operatorname{argmin}} \quad \langle\mathcal{T}(G_K),\mathcal{T}(\Delta_K)\rangle_F \quad .$ □

Unfortunately the Convex Hull argument is no more an equality in the convolution setup since:

$$\mathrm{CH}(\mathcal{T}(\mathcal{M}_{\mathrm{kernel}}) \cap \mathrm{St}) \subseteq \mathcal{T}(\mathcal{M}_{\mathrm{kernel}}) \cap \mathrm{CH}(\mathrm{St}) = \{\Delta \in \mathcal{M}_{\mathrm{kernel}} \text{ such that } \|\mathcal{T}(\Delta)\|_{op} \leq 1\}$$

The convolutional polar factor ConvPF is thus the optimum on a subspace of the unit ball, inducing an overconstraint.