Muon Implementation Based on Streaming Power Iteration: 5. Extensions

Su Jianlin

2026-04-17

The central theme of this series is "streaming power iteration". As the name suggests, it combines two components: "streaming" and "power iteration". Here, "power iteration" refers to a classical multi-step iterative method for computing the singular value decomposition (SVD) of a matrix, while "streaming" means distributing the originally multi-step algorithm across each training step, making the computational cost acceptable. The core idea is: instead of performing a complex calculation in one go, continuously approximate the target during training.

As an extension of this series, this article introduces several other applications of the "streaming" idea, further illustrating how computationally expensive operations can be cleverly integrated into the training process through streaming transformations.

1 Orthogonal Projection

In certain scenarios, we may want to enforce orthogonality constraints on specific parameter matrices. Orthogonal matrices offer good numerical stability, avoiding issues such as numerical explosion or vanishing, and can provide better theoretical guarantees in some designs. Of course, determining where it is appropriate to impose orthogonality constraints requires case-by-case analysis, which we will not elaborate on here.

In the articles "Steepest Descent on Manifolds: 2. Muon + Orthogonal" and "Steepest Descent on Manifolds: 3. Muon + Stiefel" we have explored the orthogonal (Stiefel) manifold, although that approach aimed to derive new update rules by combining steepest descent, making it more complex. Here we consider a simpler approach: reprojecting (retracting) parameters back onto the orthogonal manifold after each update step.

Without loss of generality, suppose a parameter matrix $\boldsymbol{W} \in \mathbb{R}^{n\times m}$ with $n \geq m$; the operation we need can be written as

$$\boldsymbol{W}_t = \boldsymbol{W}_{t-1} - \eta\boldsymbol{\Phi}_t \qquad \to \qquad \boldsymbol{W}_t = \text{orth}(\boldsymbol{W}_{t-1} - \eta\boldsymbol{\Phi}_t) \tag{1}$$

where orth is defined as

$$\text{orth}(\boldsymbol{W}) = \mathop{\text{argmin}}_{\boldsymbol{O}^\top\boldsymbol{O}=\boldsymbol{I}} \|\boldsymbol{W} - \boldsymbol{O}\|_F \tag{2}$$

that is, finding the orthogonal matrix closest to $\boldsymbol{W}$. This operation has already appeared in our first blog post on Muon "Appreciating the Muon Optimizer: From Vector to Matrix Essentials", and it is essentially the msign operation in Muon. Thus, the update can alternatively be written as

$$\boldsymbol{W}_t = \text{msign}(\boldsymbol{W}_{t-1} - \eta\boldsymbol{\Phi}_t) \tag{3}$$

2 Streaming Iteration

In other words, after each update step, we perform a msign operation to project the parameters back onto the orthogonal manifold. However, computing msign is neither cheap nor negligible.

While we know that msign is central to Muon, Muon runs its msign in BF16, whereas model parameters are stored in FP32, so computing msign on parameters in FP32 incurs a significant cost.

Efficient computation of msign is based on Newton-Schulz iteration, which we have discussed in detail in "Newton-Schulz Iteration for msign Operators (Part 1)" and "Newton-Schulz Iteration for msign Operators (Part 2)". Different Newton-Schulz iterations use different polynomials with varying degrees and coefficients, resulting in different convergence speeds. A classical third-order format is

$$\boldsymbol{X}_t = \frac{3}{2}\boldsymbol{X}_{t-1} - \frac{1}{2}\boldsymbol{X}_{t-1}\boldsymbol{X}_{t-1}^\top\boldsymbol{X}_{t-1}, \qquad \boldsymbol{X}_0 = \boldsymbol{W} \tag{4}$$

which provably satisfies $\lim_{t\to\infty}\boldsymbol{X}_t = \operatorname{msign}(\boldsymbol{W})$. Although classic, this iteration converges slowly; in Muon we typically use faster fifth-order versions. Regardless, full iteration incurs significant cost.

However, in the context of parameter updates, is it truly necessary to perform the complete iteration at every step? Assuming $\boldsymbol{W}_{t-1}$ is already orthogonal or nearly so, and given a small learning rate η, the updated $\boldsymbol{W}_{t-1} - \eta\boldsymbol{\Phi}_t$ will not deviate significantly from orthogonality. This is where the streaming idea becomes effective—perhaps one iteration suffices per training step. Consider instead

$$\boldsymbol{W}_t = \frac{3}{2}\tilde{\boldsymbol{W}}_t - \frac{1}{2}\tilde{\boldsymbol{W}}_t\tilde{\boldsymbol{W}}_t^\top\tilde{\boldsymbol{W}}_t, \qquad \tilde{\boldsymbol{W}}_t = \boldsymbol{W}_{t-1} - \eta\boldsymbol{\Phi}_t \tag{5}$$

Asymptotically, this achieves the same effect as (3), and requires only one additional computation $\tilde{\boldsymbol{W}}_t\tilde{\boldsymbol{W}}_t^\top\tilde{\boldsymbol{W}}_t$, which is significantly cheaper than a full Newton-Schulz iteration.

3 Spectral Constraints

Generally, orthogonal constraints can only be applied to certain special matrices, as they are too restrictive, reducing the matrix's degrees of freedom by half—effectively halving the number of free parameters. Often, one might prefer a looser spectral constraint, such as the singular value clipping mentioned in "Higher-Order MuP: A More Concise and More Refined Spectral Conditioning Scaling".

If msign sets all singular values of a matrix to 1, then singular value clipping only reduces singular values greater than 1 to 1, leaving others unchanged. In "Computing Singular Value Clipping mclip via msign (Part 1)" and "Computing Singular Value Clipping mclip via msign (Part 2)", this is referred to as mclip. If what we want is to clip singular values to lie within 1 after each parameter update, we write

$$\boldsymbol{W}_t = \operatorname{mclip}(\boldsymbol{W}_{t-1} - \eta\boldsymbol{\Phi}_t) \tag{6}$$

However, computing mclip is considerably more expensive than msign. Previous work explored using msign to compute mclip, for example via the identity

$$\operatorname{mclip}(\boldsymbol{M}) = \frac{1}{2}\Big[\boldsymbol{M} + \operatorname{msign}(\boldsymbol{M}) + (\operatorname{msign}(\boldsymbol{M}) - \boldsymbol{M})\operatorname{msign}(\boldsymbol{M}^\top\boldsymbol{M} - \boldsymbol{I})\Big] \tag{7}$$

which requires two msign computations and is thus quite costly. Alternatively, one could run an additional streaming power iteration on $\boldsymbol{W}$ to compute mclip via SVD, but that introduces another cache variable and becomes cumbersome.

4 Sequential Clipping

Let us reinterpret mclip from a different angle: mclip converts all singular values greater than 1 to 1, so a necessary step is to clip the largest singular value to 1 (if greater than 1). After clipping, if other singular values exceed 1, the largest remaining becomes the new principal singular value. This suggests that repeatedly applying "clipping the principal singular value to 1" suffices to achieve mclip.

The advantage of this "sequential clipping" strategy is that computing only the principal singular value and corresponding vectors (denote this operation as SVD1) is far cheaper than a full SVD. It can be efficiently approximated via vectorized power iteration

$$\boldsymbol{v} \leftarrow \frac{\boldsymbol{W}^\top \boldsymbol{W}\boldsymbol{v}}{\|\boldsymbol{W}^\top \boldsymbol{W}\boldsymbol{v}\|} \tag{8}$$

which converges to the right principal singular vector $\boldsymbol{v}_1$ of $\boldsymbol{W}$, from which we obtain $\sigma_1 = \|\boldsymbol{W}\boldsymbol{v}_1\|$ and $\boldsymbol{u}_1 = \boldsymbol{W}\boldsymbol{v}_1/\sigma_1$. In practice, we use a fixed number of iterations to get an approximation. Then, by maintaining the "streaming" philosophy, we perform only a single principal singular value clipping per update:

$$\boldsymbol{W}_t = \tilde{\boldsymbol{W}}_t - \max(\sigma_1 - 1, 0)\boldsymbol{u}_1\boldsymbol{v}_1^\top, \quad \sigma_1, \boldsymbol{u}_1, \boldsymbol{v}_1 = \text{SVD1}(\tilde{\boldsymbol{W}}_t), \quad \tilde{\boldsymbol{W}}_t = \boldsymbol{W}_{t-1} - \eta\boldsymbol{\Phi}_t \tag{9}$$

Over long-term training, this keeps the singular values of $\boldsymbol{W}$ near 1 (slightly above due to approximation and streaming effects). This "streaming singular value clipping" can be seen as a variant of the "Spectral Weight Decay" introduced in "From Spectral Norm Gradients to Novel Weight Decay Ideas", and is dubbed "Spectral Hammer" in the paper "Training Transformers with Enforced Lipschitz Constants".

5 Other Examples

When is the streaming idea applicable? A typical scenario is when we expect a variable to change slowly over long training periods, suggesting that minimal iterations per step may suffice to correct updates. Beyond the two new examples above, we have already employed the "streaming" idea in earlier articles—a few are recalled briefly.

In "Steepest Descent on Manifolds: 3. Muon + Stiefel" and "Steepest Descent on Manifolds: 4. Muon + Spectral Spheres", solving for steepest descent on corresponding manifolds required solving nonlinear equations. There, we mentioned fixed-point iteration; later in "Steepest Descent on Manifolds: 5. Dual Gradient Descent", we discussed a dual gradient descent solution and proposed distributing the solution process across each training step via streaming.

Similarly, in "MoE Tours: 6. Optimal Allocation Promotes Balance", we viewed MoE's load balancing from an optimal assignment perspective. Solving the dual problem originally required alternating optimization of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ to convergence at each step. However, since $\boldsymbol{\beta}$ should change only slightly between steps, the streaming idea suggests that updating $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ just once per step suffices to achieve good load balancing.

6 Summary

This article introduced two additional applications of the "streaming" idea: achieving parameter projection onto orthogonal (Stiefel) manifolds with minimal overhead, and clipping parameter spectral norms to stay at or below 1. These examples again illustrate the power of the streaming concept: many operations seemingly requiring one-off, complex computations can be decomposed into incremental adjustments integrated into each training step.