Implementing Momentum Orthogonalized by Newton-Schulz (MUON)

Erik Lidman Hillbom, Elias Lindstenz, Alve Carr, Tatsuya Hongka
eriklh@kth.se, elialin@kth.se, alvec@kth.se, hongka@kth.se
SF1672 Linjärprogrammering Grupp 13

November 18, 2025

The presentation will be conducted in English

Introduction

Artificial Intelligence (AI) and machine learning has contributed immensely to society. For example, Google's AlphaFold (Google DeepMind, n.d) predicts protein structures, thereby accelerating medical research that can save millions of lives, and Wang, X. et al. (2024) have found a way to identify cancer tumors. In machine learning, model parameters are improved during a process called training using some optimization algorithm. Our goal is to implement the MUON optimization algorithm, which is more efficient than the de facto standard Adam optimizer, to test the current frontier of AI optimization.

Optimization Problem Formulation

The optimization objective is to find parameters $\theta \in \mathbb{R}^d$ that minimize $\mathcal{R}$

$$\mathcal{R}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{P}_{\text{data}}}\big[L(h_\theta(x), y)\big] \approx \frac{1}{N}\sum_{i=1}^{N} L(h_\theta(x_i), y_i)$$

where $(x, y) \sim \mathcal{P}_{\text{data}}$ is the true data distribution with $x \in \mathcal{X}$ (the input space) and $y \in \mathcal{Y}$ (the output space). $h_\theta : \mathcal{X} \to \mathcal{Y}$ is a model parameterized by $\theta \in \mathbb{R}^d$. $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ measures the loss between the model's predicted output $\hat{y} \in \mathcal{Y}$ and the true output $y \in \mathcal{Y}$. $\{(x_i, y_i)\}_{i=1}^{N}$, is a finite dataset of N samples drawn i.i.d., $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, to approximate the true unknown data distribution.

Gradient-Based Optimization

The simplest solution to the optimization problem is the gradient descent algorithm, with the following update rule at each iteration $t \in \mathbb{N}$

$$\theta_{t+1} \leftarrow \theta_t - \eta\,\nabla\mathcal{R}(\theta_t),$$

where $\eta > 0$ is the learning rate, $\theta \in \mathbb{R}^{m\times n}$ are the model weights, and $\nabla\mathcal{R}(\theta_t)$ is the gradient of the average loss L for the parameters θ, defined as

$$\nabla\mathcal{R}(\theta_t) = \begin{bmatrix} \frac{\partial L(\theta_t)}{\partial \theta_{1,1}} & \cdots & \frac{\partial L(\theta_t)}{\partial \theta_{1,n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial L(\theta_t)}{\partial \theta_{m,1}} & \cdots & \frac{\partial L(\theta_t)}{\partial \theta_{m,n}} \end{bmatrix}.$$

In practice, a stochastic gradient estimate $g(\theta)$ is often used based on a subset of the dataset, leading to the stochastic gradient descent (SGD) update

$$\theta_{t+1} \leftarrow \theta_t - \eta\, g(\theta_t).$$

Adam is a common optimizer with the following update rule (no weight decay)

$$m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g(\theta_t),$$
$$v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g(\theta_t)^2.$$
$$\theta_{t+1} \leftarrow \theta_t - \eta \frac{m_t}{\sqrt{v_t} + \epsilon},$$

Adam scales the gradients with the first- and second-moment exponentially weighted average estimates for adaptive per-parameter learning rates. The hyperparameters $\beta_1 \in [0,1)$ and $\beta_2 \in [0,1)$ control the weighted averages, and a small $\epsilon > 0$ creates numerical stability by preventing division by zero.

Momentum Orthogonalized by Newton-Schulz

Momentum Orthogonalized by Newton-Schulz (MUON) is a new and significantly faster optimizer. Adam commonly creates low-rank gradient matrices, where only a few gradient directions dominate the update iteration. MUON equalizes the magnitudes of all directions of the gradient by orthogonalization. The creators of MUON hypothesize that by balancing the effects of update directions, the loss function is more quickly minimized. The intuition is that small derivatives are equally important. Orthogonalization is done per neural network layer (MUON generally is especially designed for linear layers) and thus scales per-parameter with the whole layer, which may be more efficient than Adam's independent per-parameter scaling.

Orthogonalization

Every matrix can be factorized as the product of two rotation matrices ($U \in \mathbb{R}^{m \times m}$ and $V^\top \in \mathbb{R}^{n \times n}$) and a diagonal scaling matrix ($\Sigma \in \mathbb{R}^{m \times n}$).

$$M = U \Sigma V^\top$$

The diagonal of Σ is called the singular values and this factorization is called Singular Value Decomposition (SVD). A matrix is semi-orthonormal if $OO^T = I$ or $O^T O = I$. U and $V^\top$ are semi-orthonormal. MUON's goal is to make the singular values all one so that the matrix M becomes orthonormal $M = UIV^\top = UV^\top$. But the naive SVD factorization is too slow ($O(n^3)$) to run each iteration. Therefore, the Newton-Schulz algorithm is used.

Newton-Schulz Algorithm

Newton-Schulz, as used in MUON, is an iterative algorithm to approximate M with its nearest semi-orthogonal matrix O.

$$Ortho(M) = argmin_O \{||O - M||_F\}$$

To map singular values to 1, one could apply the sign function (1 if greater than 0, 0 if equal to 0, and -1 if less than 0) to Σ. However, the sign function cannot be applied directly because it is discontinuous and unstable. Instead, odd polynomials are used to estimate it, since they have the nice property that they only affect singular values.

$$p(U\Sigma V^\top) = Up(\Sigma)V^\top$$

Since Σ is diagonalized, this is the same as applying a scalar polynomial on the diagonal entries of Σ. The goal is then to find an odd scalar polynomial $p(x)$ that when iterated causes Σ to converge to the identity matrix. Empirically, the creators of MUON found five iterations of $p(x) = 3.4445x + 4.7770x^3 + 2.0315x^5$ to be good. This maps singular values $s_i \in [0, 1]$ to $[0.7, 1.3]$, which empirically does not affect the results despite being an imperfect interval.

The Update Rule

For each neural network layer and iteration t of MUON, the following algorithm is completed. First, (1) a gradient estimate ($g(\theta_t)$) is calculated and (2) momentum (M_t) is computed on a subset of the data set with the model parameters (θ_{t-1}) from the previous iteration. Then (3) the momentum matrix is normalized (N_t) using the Frobenius norm since the scalar odd polynomial in the Newton-Schulz algorithm expects values $M_{t_{i,j}} \in [0, 1]$. Orthogonalization (4) is then completed by applying Newton-Schulz five times. Finally, (5) the model parameters are updated with the learning rate η.

$$\begin{align}
&\text{Gradient Estimate } g(\theta_t) \tag{1}\\
&M_t \leftarrow \beta M_{t-1} + g(\theta_t) \tag{2}\\
&N_t \leftarrow \frac{M_t}{||M_t||_F} \tag{3}\\
&O_t \leftarrow NewtonSchulz5(N_t) \tag{4}\\
&\theta_t \leftarrow \theta_t + \eta O_t \tag{5}
\end{align}$$

References

Bernstein, J. (2024) Newton-Schulz. Available at: https://docs.modula.systems/algorithms/newton-schulz/ (Accessed: 18 November 2025).

Google DeepMind (no date) AlphaFold. Available at: https://deepmind.google/science/alphafold/ (Accessed: 18 November 2025).

Huang, J.-B. (2025) This Simple Optimizer Is Revolutionizing How We Train AI [Muon]. [Video] Available at:

https://www.youtube.com/watch?v=bO5nvE289ec (Accessed: 18 November 2025).

Jordan, K., Jin, Y., Boza, V., You, J., Cesista, F., Newhouse, L. and Bernstein, J. (2024) Muon: An optimizer for hidden layers in neural networks. Available at: https://kellerjordan.github.io/posts/muon/ (Accessed: 18 November 2025).

Wang, X. et al. (2024) ‘A pathology foundation model for cancer diagnosis and prognosis prediction’, Nature, 634, pp. 970–978. Available at: https://doi.org/10.1038/s41586-024-07894-z.