Matrix-Based Optimizers

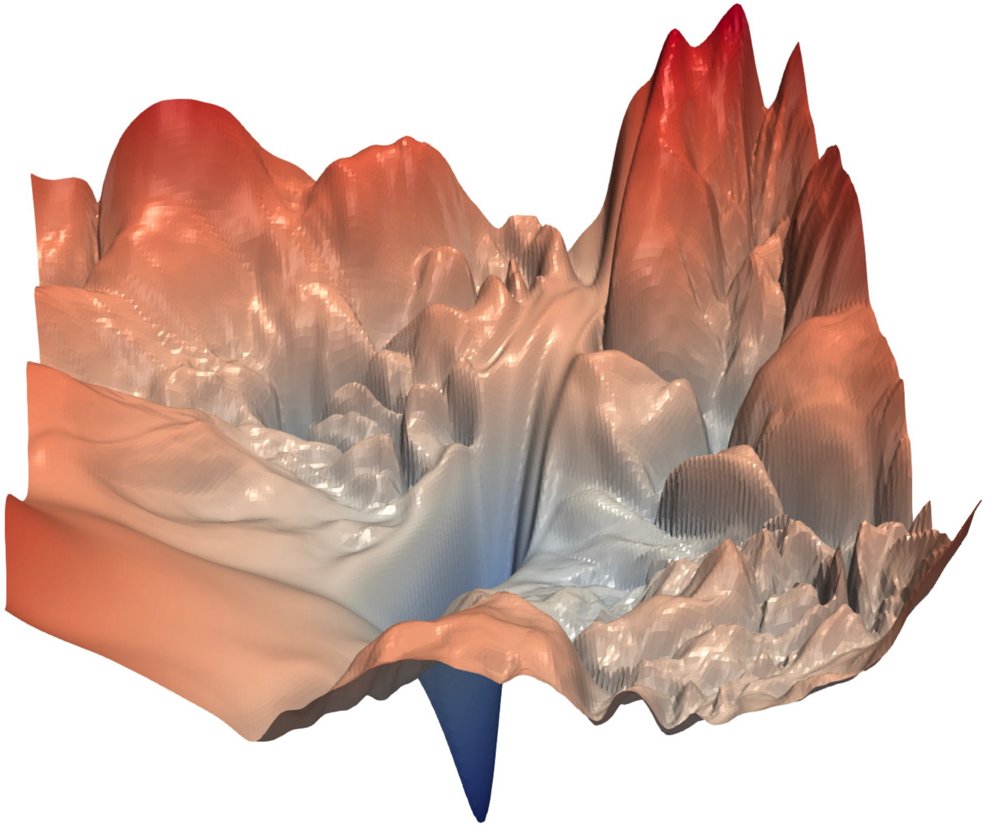
Ruizhong Qiu
October 15, 2025

Neural Architectures & Optimizers

- General form of deep learning:

$$\min_{\theta} L(\theta).$$

    - Optimize a loss L w.r.t. parameters θ.
- **Prevailing:** Architecture-agnostic optimizers.
    - Treat parameters as a single vector θ.
        - E.g., gradient descent, Adam, …
    - Only need a gradient oracle $\nabla_{\theta} L$.
        - But overlook the architecture of neural networks.
- **Emerging:** Architecture-aware optimizers.
    - Neural networks have many matrix parameters.
    - Different updates for matrices and vectors?
        - E.g., Shampoo [1,2], Muon [3,4], …
        - **Industry:** Kimi K2 was pretrained using Muon [5].



- Loss landscape of a neural network [6].

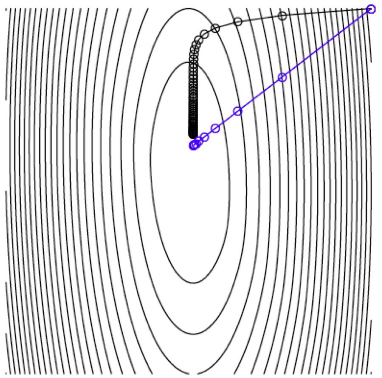
[1] Gupta et al. Shampoo: Preconditioned stochastic tensor optimization. ICML, 2018.
[2] Morwani et al. A new perspective on Shampoo's preconditioner. ICLR, 2025.
[3] Jordan et al. 2025. https://kellerjordan.github.io/posts/muon/
[4] Bernstein et al. 2025. https://jeremybernste.in/writing/deriving-muon
[5] Kimi Team. Kimi K2: Open agentic intelligence. Moonshot AI, 2025.
[6] Li et al. Visualizing the loss landscape of neural nets. NeurIPS, 2018.

Contents

Preliminaries: Gauss–Newton Method

- A slightly finer-grained assumption:
  - Model input: x. Model parameters: θ. Model output: $z = f(x, \theta)$.
  - Ground truth of x: y_x. Loss function: $\ell(y_x, z)$. Overall loss: $\mathcal{L}(\theta) = \mathbb{E}_x\big[\ell\big(y_x, f(x, \theta)\big)\big]$.
  - Finding a small update $\Delta\theta$ that approximately minimizes $\mathcal{L}(\theta - \Delta\theta)$:

$$\mathcal{L}(\theta - \Delta\theta) \approx \mathcal{L}(\theta) - \nabla_\theta \mathcal{L}(\theta)^\top \Delta\theta + \frac{1}{2}\Delta\theta^\top \nabla_\theta^2 \mathcal{L}(\theta) \Delta\theta.$$



- Handling the second-order term in Taylor approximation:
  - Gradient descent (simply dropping the Hessian):

$$\mathcal{L}(\theta - \Delta\theta) \approx \mathcal{L}(\theta) - \nabla_\theta \mathcal{L}(\theta)^\top \Delta\theta \quad \Rightarrow \quad \Delta\theta \propto -\nabla_\theta \mathcal{L}(\theta).$$

  - Newton–Raphson method (exact Hessian; computationally expensive):

$$\Delta\theta \approx \nabla_\theta^2 \mathcal{L}(\theta)^{-1} \nabla_\theta \mathcal{L}(\theta), \qquad \nabla_\theta^2 \mathcal{L}(\theta) = \mathbb{E}_x\big[\nabla_\theta f(x,\theta)^\top \nabla_z^2 \ell\big(y_x, f(x,\theta)\big) \nabla_\theta f(x,\theta)\big] + \mathbb{E}_x\big[\nabla_\theta^2 f(x,\theta)\big] \times_z \nabla_\theta \mathcal{L}(\theta).$$

  - Gauss–Newton method (because the first term usually dominates in deep learning [1]):

$$\nabla_\theta^2 \mathcal{L}(\theta) \approx \mathbb{E}_x\big[\nabla_\theta f(x,\theta)^\top \nabla_z^2 \ell\big(y_x, f(x,\theta)\big) \nabla_\theta f(x,\theta)\big].$$

    - This matrix is called the **Gauss–Newton component** (GN) of the Hessian.
    - However, it is still expensive to compute the GN if the output dimensionality is large (e.g., LLMs).

[1] Sankar et al. A deeper look at the Hessian eigenspectrum of deep neural networks and its applications to regularization. AAAI, 2021.

Shampoo: Kronecker Product Approximation

- Shampoo [1] is an optimizer that approximates the GN for matrix parameters.
  - Shampoo ignores the interaction between layers and only considers the GN for each matrix W.
    - Reducing the problem to approximating $\nabla_{\mathrm{vec}(W)} f(x,\theta)^\top \nabla_z^2 \ell(y_x, f(x,\theta)) \nabla_{\mathrm{vec}(W)} f(x,\theta)$.
  - For the softmax cross entropy loss $\ell(y,z) = -\log \mathrm{softmax}(z)_y$, the GN of $\mathrm{vec}(W)$ is [2]:
    $$\mathbb{E}_x\left[\nabla_{\mathrm{vec}(W)} f(x,\theta)^\top \nabla_z^2 \ell(y_x, f(x,\theta)) \nabla_{\mathrm{vec}(W)} f(x,\theta)\right] = \mathbb{E}_{x,\hat{y}\sim\mathrm{softmax}(f(x,\theta))}\left[\mathrm{vec}(G_{x,\hat{y}})\,\mathrm{vec}(G_{x,\hat{y}})^\top\right],$$
    - where $G_{x,\hat{y}} := \nabla_W L(\hat{y}, f(x,\theta))$ denotes the matrix-shape gradient of W.
  - Furthermore, we can upper-bound the GN via a Kronecker product: If $\mathrm{rank}(G_{x,\hat{y}}) \le r$, then [1]:
    $$\mathbb{E}_{x,\hat{y}\sim\mathrm{softmax}(f(x,\theta))}\left[\mathrm{vec}(G_{x,\hat{y}})\,\mathrm{vec}(G_{x,\hat{y}})^\top\right]$$
    $$\preccurlyeq r\left(\mathbb{E}_{x,\hat{y}\sim\mathrm{softmax}(f(x,\theta))}\left[G_{x,\hat{y}} G_{x,\hat{y}}^\top\right]\right)^{\frac{1}{4}} \otimes \left(\mathbb{E}_{x,\hat{y}\sim\mathrm{softmax}(f(x,\theta))}\left[G_{x,\hat{y}}^\top G_{x,\hat{y}}\right]\right)^{\frac{1}{4}}.$$
    - Here, $\preccurlyeq$ denotes the Loewner order, and $\otimes$ denotes the Kronecker product.

[1] Gupta et al. Shampoo: Preconditioned stochastic tensor optimization. ICML, 2018.
[2] Morwani et al. A new perspective on Shampoo's preconditioner. ICLR, 2025.

Shampoo: Matrix Update

- Shampoo uses this Kronecker product approximation to derive matrix updates.

  - To reduce computation, Shampoo uses y_x instead of sampling $\hat{y} \sim \mathrm{softmax}\big(f(x,\theta)\big)$.
  $$L \coloneqq \mathbb{E}_x\left[G_{x,y_x} G_{x,y_x}^{\top}\right], \qquad R \coloneqq \mathbb{E}_x\left[G_{x,y_x}^{\top} G_{x,y_x}\right], \qquad G \coloneqq \mathbb{E}_x\left[G_{x,y_x}\right] = \nabla_W \mathcal{L}(\theta).$$
    - L and R are called **preconditioners**.

  - Then, the upper bound is $\propto L^{1/4} \otimes R^{1/4}$. The vectorized form of the matrix update ΔW:
  $$\mathrm{vec}(\Delta W) \propto \left(L^{1/4} \otimes R^{1/4}\right)^{-1} \mathrm{vec}(G) = \left(L^{-1/4} \otimes R^{-1/4}\right) \mathrm{vec}(G) = \mathrm{vec}\left(L^{-1/4} G R^{-(1/4)\top}\right) = \mathrm{vec}\left(L^{-1/4} G R^{-1/4}\right).$$

  - Rewriting it into the matrix form yields the Shampoo update:
  $$\Delta W \propto L^{-1/4} G R^{-1/4}.$$

[1] Gupta et al. Shampoo: Preconditioned stochastic tensor optimization. ICML, 2018.

Shampoo/SOAP: Practical Variants

- Preconditioner momentum [1]: for a hyperparameter $0 < \beta_1 < 1$,
$$L \leftarrow \beta_1 L + (1-\beta_1) G_{x,y_x} G_{x,y_x}^\top, \qquad R \leftarrow \beta_1 R + (1-\beta_1) G_{x,y_x}^\top G_{x,y_x}.$$

- Matrix gradient momentum [2]: for a hyperparameter $0 < \beta_2 < 1$,
$$G \leftarrow \beta_2 G + (1-\beta_2) G_{x,y_x}.$$

- Exponent other than $1/4$ [2]: for a hyperparameter $0 < \beta_3 < 1$,
$$\Delta W \propto L^{-\beta_3} G R^{-\beta_3}.$$

- SOAP: Generalizing Adam to both the entries and the eigenvalues in Shampoo [2].

[1] Gupta et al. Shampoo: Preconditioned stochastic tensor optimization. ICML, 2018.
[2] Vyas et al. SOAP: Improving and stabilizing Shampoo using Adam for language modeling. ICLR, 2025.

Empirical Performance

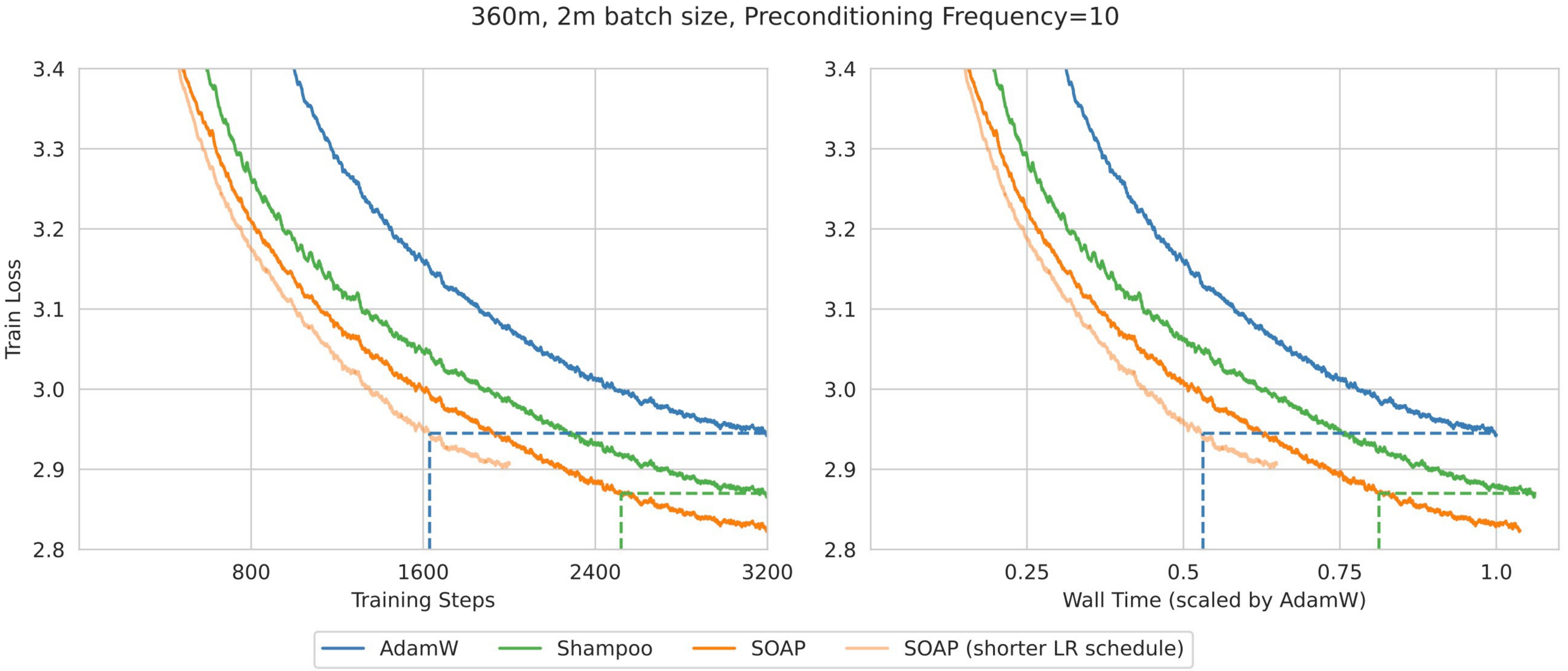


Figure 3: Comparing performance of tuned runs for AdamW, Shampoo (using DistributedShampoo (Shi et al., 2023) implementation) and SOAP. Shampoo and SOAP use preconditioning frequency of 10. We observe a $\geq 40\%$ reduction in the number of iterations and a $\geq 35\%$ reduction in wall clock time compared to AdamW, and approximately a 20% reduction in both metrics compared to Shampoo. See Figure 1 for 660m results, Sections 6.2 and 6.3 for ablations of preconditioning frequency and batch size respectively, and Section 5 for detailed calculation of efficiency improvement and experimental methodology.

[1] Vyas et al. SOAP: Improving and stabilizing Shampoo using Adam for language modeling. ICLR, 2025.

Contents

From Shampoo to Muon

- Suppose $W \in \mathbb{R}^{m\times n}$. Without any momentum in Shampoo,

$$L \coloneqq GG^\top \in \mathbb{R}^{m\times m}, \qquad R \coloneqq G^\top G \in \mathbb{R}^{n\times n}, \qquad G \coloneqq \nabla_W \mathcal{L}(\theta) \in \mathbb{R}^{m\times n}.$$

- The singular value decomposition (SVD) of G:

$$G =: U\Sigma V^\top,$$

  - where $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$ are column-orthonormal, and $\Sigma \in \mathbb{R}^{r\times r}$ is diagonal with positive diagonal entries.

- Simplifying the Shampoo update yields the **Muon update**:

$$\Delta W \propto L^{-1/4} G R^{-1/4} = (U\Sigma V^\top V\Sigma U^\top)^{-1/4} U\Sigma V^\top (V\Sigma U^\top U\Sigma V^\top)^{-1/4} = UV^\top = \mathrm{msign}(G),$$

  - where msign denotes the matrix sign operation.
  - It can be shown that msign does not change no matter how we choose singular vectors.

Muon: Steepest Descent Perspective

- An alternative & more general perspective: steepest descent.
  - When updating a matrix parameter W to $W - \Delta W$:
    - The loss approximately decreases by $\langle \nabla_W \mathcal{L}(\theta), \Delta W \rangle$:
$$\mathcal{L}(\theta) - \mathcal{L}(\theta - \Delta W) \approx \langle \nabla_W \mathcal{L}(\theta), \Delta W \rangle = \mathrm{Tr}(\nabla_W \mathcal{L}(\theta)^\top \Delta W)\,.$$
    - Given layer input $v \in \mathbb{R}^n$, the layer output $(W - \Delta W)v$ deviates Wv by at most $\|\Delta W\|_2 \|v\|_2$:
$$\|(W - \Delta W)v - Wv\|_2 = \|\Delta W v\|_2 \leq \|\Delta W\|_2 \|v\|_2.$$
      - Here, $\|\Delta W\|_2$ denotes the spectral norm of ΔW, and $\|v\|_2$ denotes the Euclidean norm of v.
      - Deviating too much can fail the first-order approximation of the loss.
  - Hence, we can solve the following steepest descent problem to find the best update ΔW:
$$\max_{\|\Delta W\|_2 \leq \eta} \langle \nabla_W \mathcal{L}(\theta), \Delta W \rangle\,,$$
    - where $\eta > 0$ is the learning rate.
  - The optimal update ΔW is proportional to the matrix sign of $\nabla_W \mathcal{L}(\theta)$:
$$\Delta W = \eta\, \mathrm{msign}\big(\nabla_W \mathcal{L}(\theta)\big) = \eta\, \mathrm{msign}(G)\,.$$
    - Again, we have derived the Muon update but from another perspective.

[1] Bernstein et al. 2025. https://jeremybernste.in/writing/deriving-muon
[2] Bernstein & Newhouse. Old optimizer, new norm: An anthology. OPT, 2024.

Muon: Newton–Schulz Iterations

- How to compute msign efficiently?
  - SVD: computationally expensive for large parameter matrices in LLMs.
  - Newton–Schulz iterations: easy to compute & fast convergence.
- Preliminaries: SVD commutes with odd polynomials.
  - Odd polynomial $p(G)\colon \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$:
    $$p(G) = p_1 G + p_3 G G^\top G + p_5 G G^\top G G^\top G + \cdots, \qquad p_1, p_3, p_5, \ldots \in \mathbb{R}.$$
  - Given SVD $G = U\Sigma V^\top$, we have $p(G) = Up(\Sigma)V^\top$ for any odd polynomial p.
- Newton–Schulz iterations:
  - When $p_1 \approx 3.4445$, $p_3 \approx -4.7750$, $p_5 \approx 2.0315$, we have
    $$\lim_{t\to\infty} p^{\circ t}(s) \in [0.7,1.3]\,\mathrm{sign}(s), \qquad s \in [-1,1].$$
  - Newton–Schulz iterations compute msign efficiently by repeatedly applying this polynomial:
    $$\lim_{t\to\infty} p^{\circ t}\left(\frac{G}{\|G\|_{\mathrm{F}}}\right) \approx \mathrm{msign}(G).$$
    - The convergence is very fast. Muon uses only 5 iterations.
    - The initial denominator $\|G\|_{\mathrm{F}}$ is to ensure numerical stability and to speed up convergence.

[1] Bernstein & Newhouse. Old optimizer, new norm: An anthology. OPT, 2024.

Muon: Practical Variants

- A number of practical variants:
  - Applying Muon to hidden layers only and AdamW to input and output layers [1].
  - Nesterov-style momentum instead of ordinary momentum [1].
  - Separating Q, K, & V projections although the common implementation is a single matrix [1].
  - Rescaling the RMS norm [2] of matrix updates to 0.2 (that of typical AdamW updates) [3].
  - MuonClip: Rescaling Q & K projections after each Muon update if the RMS norm is too large [4].
  - AdaMuon: Generalizing Adam to Muon [5].

[1] Jordan et al. 2025. https://kellerjordan.github.io/posts/muon/
[2] Bernstein et al. 2025. https://jeremybernste.in/writing/deriving-muon
[3] Liu et al. Muon is scalable for LLM training. Moonshot AI, 2025.
[4] Kimi Team. Kimi K2: Open agentic intelligence. Moonshot AI, 2025.
[5] Si et al. AdaMuon: Adaptive Muon optimizer.

Empirical Performance

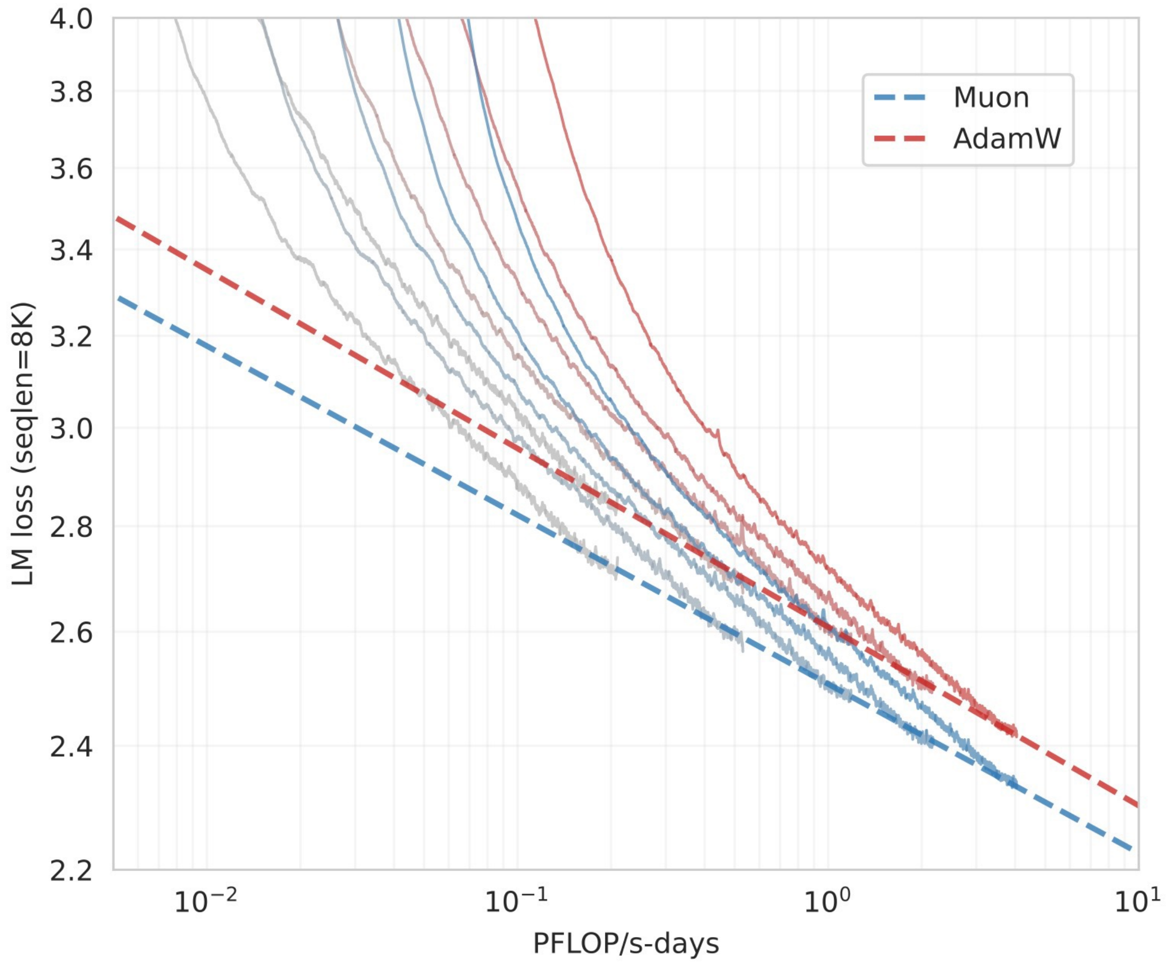


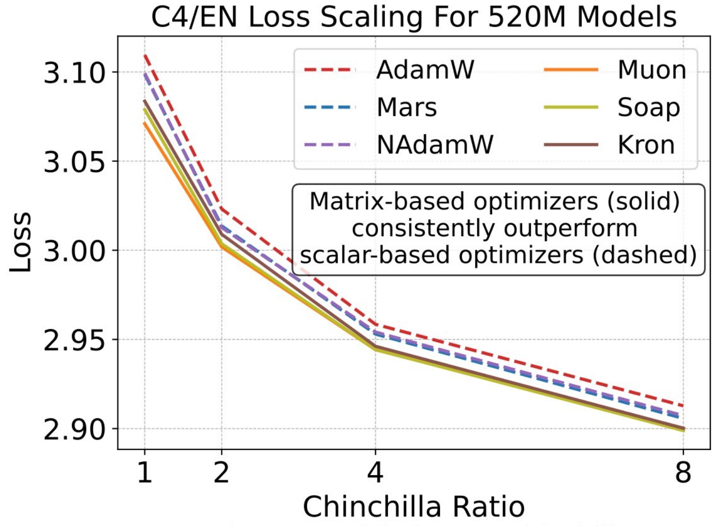
Figure 3: Fitted scaling law curves for Muon and AdamW optimizers.

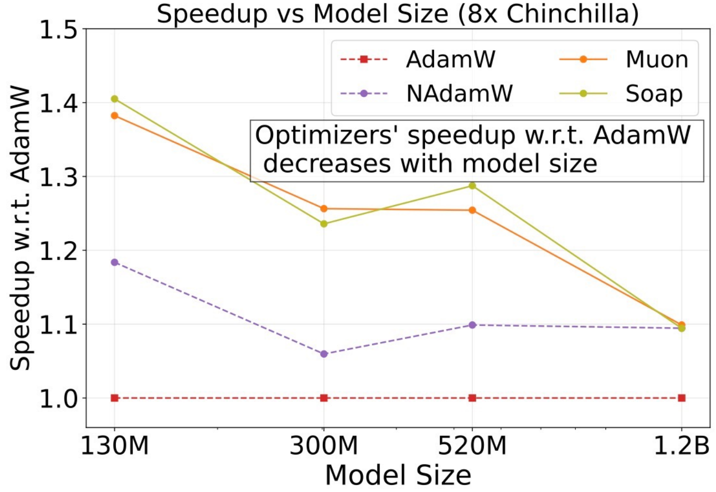
[1] Liu et al. Muon is scalable for LLM training. Moonshot AI, 2025.

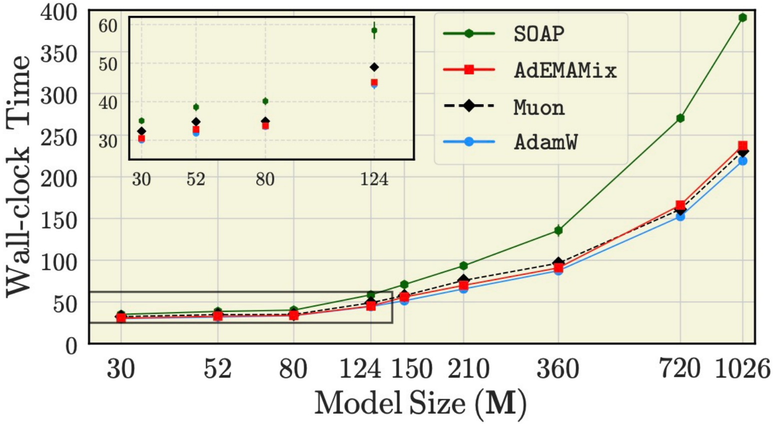
Contents

Convergence vs Scale

- Matrix-based optimizers consistently outperform scalar-based optimizers across Chinchilla ratios [1].
- The speedup over AdamW decreases as the model size increases under respective optimal hyperparameters [1].
- Muon has comparable efficiency with AdamW while SOAP is less scalable [2].







[1] Wen et al. Fantastic pretraining optimizers and where to find them. 2025.
[2] Semenov et al. Benchmarking optimizers for LLM pretraining. 2025.

Thanks!