CORRECTION OF DECOUPLED WEIGHT DECAY: DISENTANGLE EFFECTIVE LEARNING RATE AND WEIGHT NORM CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Since AdamW, decoupled weight decay has replaced L_2 regularization as the standard weight control method. More recently, steady-state analysis has suggested that decoupled weight decay should be proportional to learning rate squared to maintain stable weight norm, but such analysis can be derived from either orthogonality or independence of the weight and the gradient. We show that when applied to Muon / Scion, weight-gradient independence implies that the effective learning rate is momentum-dependent, a result that would be hard to explain with weight-gradient orthogonality alone. We show that optimal effective learning rate transfers across momentum values and enables log-time momentum scheduling for improved LLM training. Furthermore, models exhibit aggregated near-scale-invariance which explains the performance difference and altered model training dynamics due to weight decay correction. We conclude with both practical recommendations and implications of our results.

1 INTRODUCTION

L_2 regularization, a common technique for controlling model weight growth and preventing overfitting, is equivalent to weight decay for unmodified SGD. For adaptive gradient methods such as SGD with momentum (Sutskever et al., 2013) and Adam (Kingma & Ba, 2015), weight decay is no longer equivalent to L_2 regularization, and empirical observations have led to the development of the decoupled weight decay of AdamW (Loshchilov & Hutter, 2019) that outperforms the original Adam with the following update rules:

$$\begin{aligned}
\boldsymbol{g}_t &\leftarrow \nabla_\theta f_t(\boldsymbol{\theta}_{t-1}) \\
\boldsymbol{m}_t &\leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t \\
\boldsymbol{v}_t &\leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2 \\
\boldsymbol{u}_t &\leftarrow \frac{\boldsymbol{m}_t/(1-\beta_1^t)}{\sqrt{\boldsymbol{v}_t/(1-\beta_2^t)}+\epsilon} \\
\boldsymbol{\theta}_t &\leftarrow \boldsymbol{\theta}_{t-1} - \gamma\left(\lambda\boldsymbol{\theta}_{t-1} + \boldsymbol{u}_t\right)
\end{aligned}$$

where squaring and division are understood to be element-wise, θ_t and f_t are the model weights and loss function, m_t and v_t are the first and second moments of the loss gradient g_t, u_t is the parameter update, and learning rate γ, weight decay coefficient λ, betas (β_1, β_2) and epsilon ϵ are the hyperparameters. Accordingly, we get the following expression for the expected value of the l^2-norm squared of the layer weight vectors:

$$\begin{aligned}
\mathbb{E}[||\boldsymbol{\theta}_t||^2] &= \mathbb{E}[||(1-\gamma\lambda)\boldsymbol{\theta}_{t-1} - \gamma\boldsymbol{u}_t||^2] \\
&= \mathbb{E}[(1-\gamma\lambda)^2||\boldsymbol{\theta}_{t-1}||^2 + \gamma^2||\boldsymbol{u}_t||^2 - 2\gamma(1-\gamma\lambda)\langle\,\boldsymbol{\theta}_{t-1}, \boldsymbol{u}_t\rangle] \qquad (1)
\end{aligned}$$

Kosson et al. (2024) argues that the changes of model weights can be modeled as random walk and at steady state. If we assume $\beta_1 = 0$ for simplicity, $\mathbb{E}[||\boldsymbol{u}_t||^2]$ becomes a time-independent constant C as $t \to \infty$, and $\mathbb{E}[\langle\,\boldsymbol{\theta}_{t-1}, \boldsymbol{u}_t\rangle] = 0$ since $\boldsymbol{\theta}_{t-1}$ and $\boldsymbol{u}_t$ are independent, then

$$\mathbb{E}[||\boldsymbol{\theta}_t||^2] = \mathbb{E}[(1-\gamma\lambda)^2||\boldsymbol{\theta}_{t-1}||^2 + \gamma^2 C]$$

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.

Algorithm 1 "Renormalized" AdamW

Input: Initial values $\boldsymbol{\theta}_{0,l}$ for all layers l,
Input: scheduled learning rate γ_t, weight-decay coefficientλ, (β_1, β_2), ϵ
$\boldsymbol{v}_{0,l} = \boldsymbol{m}_{0,l} = 0$
for $t = 1$ **to** T **do**
  for layer $l = 0$ **to** L **do**
    $\boldsymbol{g}_{t,l} = \nabla_{\theta_l} f_t(\boldsymbol{\theta}_{t-1,l}, \boldsymbol{\zeta}_t)$ ▷ Minibatch gradient
    $\boldsymbol{m}_{t,l} = \beta_1 \boldsymbol{m}_{t-1,l} + (1-\beta_1)\boldsymbol{g}_{t,l}$
    $\boldsymbol{v}_{t,l} = \beta_2 \boldsymbol{v}_{t-1,l} + (1-\beta_2)\boldsymbol{g}^2_{t,l}$
    $\boldsymbol{u}_{t,l} = \frac{\boldsymbol{m}_{t,l}/(1-\beta_1^t)}{\sqrt{\boldsymbol{v}_{t,l}/(1-\beta_2^t)}+\epsilon}$
    $\boldsymbol{\theta}_{t-1,l} = \boldsymbol{\theta}_{t-1,l} - \gamma_t \lambda \boldsymbol{\theta}_{t-1,l}$
    $\boldsymbol{\theta}_{t,l} = \boldsymbol{\theta}_{t-1,l} - \gamma_t \boldsymbol{u}_{t,l}$ ▷ Standard Adam update
    if $||\boldsymbol{\theta}_{t-1,l}|| \geq \epsilon$ **then**
      $u_{t,l\|} = \frac{\langle \boldsymbol{\theta}_{t-1,l}, \boldsymbol{u}_{t,l} \rangle}{||\boldsymbol{\theta}_{t-1,l}||}$
      $\boldsymbol{\theta}_{t,l} = \frac{|||\boldsymbol{\theta}_{t-1,l}|| - \gamma_t u_{t,l\|}|}{||\boldsymbol{\theta}_{t,l}||+\epsilon} \boldsymbol{\theta}_{t,l}$ ▷ Only keep the contribution of $u_{t,l\|}$ to the norm
    end if
  end for
end for

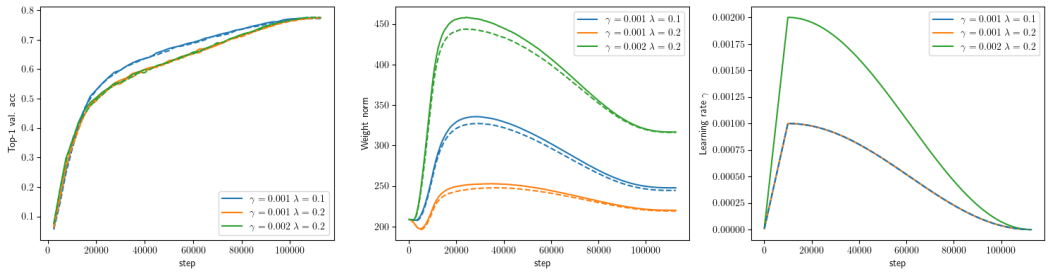


Figure 1: Training a ViT-S/16 with "Renormalized" AdamW results in negligible differences in top-1 val. accuracy (77.15 vs. 77.45 for the $\gamma = 0.001$, $\lambda = 0.1$ AdamW baseline), weight norm, and gradient norm throughout the training process. Notice the suppression of weight norm towards the end of the cosine learning rate decay, characteristic of AdamW. Except using the PyTorch Inception crop with crop scale lower bound $a_{min} = 0.2$, the setup is identical to Beyer et al. (2022).

At steady state $\mathbb{E}[||\boldsymbol{\theta}_t||^2] = \mathbb{E}[||\boldsymbol{\theta}_{t-1}||^2]$, we can solve for $\mathbb{E}[||\boldsymbol{\theta}_t||^2]$:

$$\mathbb{E}[||\boldsymbol{\theta}_t||^2] = \frac{\gamma C}{\lambda(2-\gamma\lambda)} \approx \frac{\gamma C}{2\lambda} \tag{2}$$

Alternatively, if the weight and the update are approximately orthogonal $\langle \boldsymbol{\theta}_{t-1}, \boldsymbol{u}_t \rangle \approx 0$ due to normalization layers, the same steady-state weight norm also follows (Defazio, 2025). If we observe $\mathbb{E}[||\boldsymbol{\theta}_t||^2] \propto \frac{\gamma}{\lambda}$, which is the right explanation? Consider the "Renormalized" AdamW optimizer above (Algorithm 1) which only keeps the contribution of the scalar projection of the update $u_{t\|} = \frac{\langle \boldsymbol{\theta}_{t-1}, \boldsymbol{u}_t \rangle}{||\boldsymbol{\theta}_{t-1}||}$ to the weight norm. If weight-update orthogonality is the explanation, we expect the change to significantly alter the training dynamics. To the contrary, we find the metrics almost unaffected (Fig. 1), lending support to the random walk model (Appx. B.1). In the following sections, we further apply the random walk model to the analysis of Muon / Scion (Jordan et al., 2024; Pethick et al., 2025) and derive its momentum-dependent effective learning rate and relative learning rate based on the hypothesis of aggregated near-scale-invariance. We validate them with ViT and modded-NanoGPT experiments and compare the performance of Scion with corrected weight decay (ScionC) and the baseline without. We conclude with related works, practical recommendations, and implications of our results.

Table 1: Norms and the associated lmos as normalized in our experiments. Sign, Spectral, and RowNorm assume matrix weight $\boldsymbol{\theta}_l = \boldsymbol{A} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ while Bias assumes vector weight $\boldsymbol{\theta}_l = \boldsymbol{b}_\ell \in \mathbb{R}^{d_{\text{out}}}$. $\boldsymbol{U}\boldsymbol{V}^\top$ refers to the reduced SVD of the input matrix with unitary matrices $\boldsymbol{U}$ and $\boldsymbol{V}^\top$ from the full SVD $\boldsymbol{A} = \boldsymbol{U}\mathrm{diag}(\boldsymbol{\sigma})\boldsymbol{V}^\top$ while $\|\boldsymbol{A}\|_{\mathcal{S}_\infty} = \max(\boldsymbol{\sigma})$ is the spectral norm of the matrix.

	Sign	Spectral	RowNorm	Bias
Norm	$d_{\text{in}} \max_{i,j} \lvert A_{i,j} \rvert$	$\sqrt{\frac{d_{\text{in}}}{d_{\text{out}}}} \lVert \boldsymbol{A} \rVert_{\mathcal{S}_\infty}$	$\max_i \sqrt{d_{\text{in}}} \lVert \mathrm{row}_i(A) \rVert_2$	RMS
LMO	$\boldsymbol{A} \mapsto -\frac{\mathrm{sign}(\boldsymbol{A})}{d_{\text{in}}}$	$\boldsymbol{A} \mapsto -\sqrt{\frac{d_{\text{out}}}{d_{\text{in}}}} \boldsymbol{U}\boldsymbol{V}^\top$	$\mathrm{row}_i(A) \mapsto -\frac{1}{\sqrt{d_{\text{in}}}} \frac{\mathrm{row}_i(A)}{\lVert \mathrm{row}_i(A) \rVert_2}$	$\boldsymbol{b}_\ell \mapsto -\frac{\boldsymbol{b}_\ell}{\lVert \boldsymbol{b}_\ell \rVert_{\text{RMS}}}$

2 Scion with corrected weight decay

2.1 Constrained Scion

As formulated in Pethick et al. (2025), the constrained variant of Scion can be considered a collection of optimizers with the following unified update rules. Given layer l and layer weight $\boldsymbol{\theta}_{t,l}$ at time $t-1$, the choice of linear minimization oracle lmo_l, momentum α, learning rate γ, and radius ρ_l:

$$\begin{aligned} \boldsymbol{g}_{t,l} &\leftarrow \nabla_{\theta_l} f_t(\boldsymbol{\theta}_{t-1,l}, \zeta_t) \\ \boldsymbol{m}_{t,l} &\leftarrow (1-\alpha)\boldsymbol{m}_{t-1,l} + \alpha \boldsymbol{g}_{t,l} \\ \boldsymbol{\theta}_{t,l} &\leftarrow (1-\gamma)\boldsymbol{\theta}_{t-1,l} + \gamma\rho_l \,\mathrm{lmo}_l(\boldsymbol{m}_{t,l}) \end{aligned}$$

Table 1 lists the lmos and the norms from which they are derived that we use in our experiments. Conceptually, we choose the norms of the layers based on the shape of the weight and their functions in the model, and lmos are the updates with unit norms in the direction of the steepest descent. Notably, the Spectral lmo is equivalent to the Muon optimizer (Jordan et al., 2024). Although equivalent up to reparameterization, the original formulation of Scion deviates significantly from the conventional terminology and makes it difficult to reason about the role of decoupled weight decay in its update rules. We therefore reformulate constrained Scion in terms of independent weight decay coefficient $\eta = \gamma$, layer-wise learning rate $\gamma_l = \gamma\rho_l$, and layer-wise weight decay coefficient $\lambda_l = \frac{1}{\rho_l}$. The update rules then become

$$\begin{aligned} \boldsymbol{g}_{t,l} &\leftarrow \nabla_{\theta_l} f_t(\boldsymbol{\theta}_{t-1,l}, \zeta_t) \\ \boldsymbol{m}_{t,l} &\leftarrow (1-\alpha)\boldsymbol{m}_{t-1,l} + \alpha \boldsymbol{g}_{t,l} \\ \boldsymbol{\theta}_{t,l} &\leftarrow (1-\eta)\boldsymbol{\theta}_{t-1,l} + \gamma_l \,\mathrm{lmo}_l(\boldsymbol{m}_{t,l}) \\ &= \boldsymbol{\theta}_{t-1,l} + \gamma_l \left(-\lambda_l \boldsymbol{\theta}_{t-1,l} + \mathrm{lmo}_l(\boldsymbol{m}_{t,l})\right) \end{aligned}$$

2.2 Momentum with normalized update

So far we have assumed no momentum and $\mathbb{E}[\langle \boldsymbol{\theta}_{t-1}, \boldsymbol{u}_t \rangle] = 0$ which implies $\mathbb{E}[\langle \boldsymbol{u}_{t-1}, \boldsymbol{u}_t \rangle] = 0$ at steady state. Qualitatively, the relationship $\mathbb{E}[||\boldsymbol{\theta}_t||^2] \propto \frac{\gamma C}{2\lambda}$ holds regardless of momentum since as $\boldsymbol{m}_{t-k,l}$ component of $\boldsymbol{m}_{t,l}$ decays, the update of the far past eventually becomes independent of the current update:

$$\lim_{k\to\infty} \mathbb{E}[\langle \boldsymbol{u}_{t-k}, \boldsymbol{u}_t \rangle] = 0$$

In fact, if the minibatch gradients $\boldsymbol{g}_t$ become independent with time-independent expected norm at steady state, the second momentum $\boldsymbol{v}_t$ of AdamW stays approximately constant, so the Total Update Contribution (TUC) of the minibatch gradients also remains constant regardless of β_1 as postulated in Kosson et al. (2024) (Appx. E). The lmos of Scion normalize the updates so the same reasoning no longer applies and we need to derive $\mathbb{E}[\langle \boldsymbol{\theta}_{t-1}, \boldsymbol{u}_t \rangle]$. Assume that the minibatch gradients become independent with time-independent expected L_2 norm C' at steady state, $\mathbb{E}[\langle \boldsymbol{g}_{t'}, \boldsymbol{g}_t \rangle] = C'^2 \delta_{t't}$,

where δ_{ij} is the Kronecker delta function. Then at steady state

$$\mathbb{E}[||\boldsymbol{m}_t||_2^2] = \mathbb{E}[||\boldsymbol{m}_{t-1}||_2^2] = (1-\alpha)^2\mathbb{E}[||\boldsymbol{m}_{t-1}||_2^2] + \alpha^2 C'^2 = \frac{\alpha}{2-\alpha}C'^2$$

$$\boldsymbol{m}_t = (1-\alpha)^k \boldsymbol{m}_{t-k} + \alpha\sum_{i=0}^{k-1}(1-\alpha)^i \boldsymbol{g}_{t-i}, \text{ so}$$

$$\mathbb{E}[\langle \boldsymbol{m}_{t-k}, \boldsymbol{m}_t\rangle] = \frac{\alpha}{2-\alpha}C'^2(1-\alpha)^k \text{ for } k \geq 1$$

$||\boldsymbol{m}_t||_2$ depends on α but TUC of each minibatch gradient $\boldsymbol{g}_t$ would stay constant if lmo doesn't normalize the update. So if lmo normalizes the update, the TUC will be $\propto \mathbb{E}[||\boldsymbol{m}_t||_2^{-1}]$ with effective learning rate $\gamma_{\text{eff}} := \gamma\sqrt{\frac{2-\alpha}{\alpha}}$. For example, consider the Bias lmo_{b_ℓ} in Table 1 that normalizes the update $\boldsymbol{u}_t = -\mathrm{lmo}_{b_\ell}(\boldsymbol{m}_t) = \frac{\boldsymbol{m}_t}{\|\boldsymbol{m}_t\|_{\text{RMS}}}$. Then

$$\mathbb{E}[\langle \boldsymbol{u}_{t-k}, \boldsymbol{u}_t\rangle] = \mathbb{E}[\frac{\langle \boldsymbol{m}_{t-k}, \boldsymbol{m}_t\rangle}{\|\boldsymbol{m}_{t-k}\|_{\text{RMS}}\|\boldsymbol{m}_t\|_{\text{RMS}}}]$$

Assume that at steady state $||\boldsymbol{m}_t||_2 \approx \mathbb{E}[||\boldsymbol{m}_t||_2]$. Then

$$\mathbb{E}[\langle \boldsymbol{u}_{t-k}, \boldsymbol{u}_t\rangle] \approx d_{\text{out}}\frac{2-\alpha}{\alpha C'^2}\mathbb{E}[\langle \boldsymbol{m}_{t-k}, \boldsymbol{m}_t\rangle] = d_{\text{out}}(1-\alpha)^k$$

We again denote the L_2 norm of the update as $||\boldsymbol{u}_t||_2 = \sqrt{d_{\text{out}}} = C$. Given $\mathbb{E}[\langle \boldsymbol{u}_{t-k}, \boldsymbol{u}_t\rangle] \approx C^2(1-\alpha)^k$:

$$\begin{aligned}
\boldsymbol{\theta}_t &= (1-\eta)\boldsymbol{\theta}_{t-1} - \gamma\boldsymbol{u}_t \\
&= -\gamma\sum_{i=0}^{\infty}(1-\eta)^i\boldsymbol{u}_{t-i} \\
\boldsymbol{\theta}_{t-1} &= -\gamma\sum_{i=0}^{\infty}(1-\eta)^i\boldsymbol{u}_{t-1-i} \\
\mathbb{E}[\langle \boldsymbol{\theta}_{t-1}, \boldsymbol{u}_t\rangle] &= -\gamma\sum_{i=0}^{\infty}(1-\eta)^i\mathbb{E}[\langle \boldsymbol{u}_{t-1-i}, \boldsymbol{u}_t\rangle] \\
&= -\gamma C^2\sum_{i=0}^{\infty}(1-\eta)^i(1-\alpha)^{i+1} \\
&= -\gamma C^2(1-\alpha)\sum_{i=0}^{\infty}(1-\eta)^i(1-\alpha)^i \\
&= -\frac{\gamma C^2(1-\alpha)}{1-(1-\eta)(1-\alpha)} = -\frac{\gamma C^2(1-\alpha)}{\eta+\alpha-\alpha\eta}
\end{aligned}$$

Recall Eq. 1 with the independent weight decay coefficient $\eta = \gamma\lambda$:

$$\mathbb{E}[||\boldsymbol{\theta}_t||^2] = \mathbb{E}[(1-\eta)^2||\boldsymbol{\theta}_{t-1}||^2 + \gamma^2||\boldsymbol{u}_t||^2 - 2\gamma(1-\eta)\langle \boldsymbol{\theta}_{t-1}, \boldsymbol{u}_t\rangle]$$

With $||\boldsymbol{u}_t||^2 = C^2$ and the expression above, at steady state $\mathbb{E}[||\boldsymbol{\theta}_t||^2] = \mathbb{E}[||\boldsymbol{\theta}_{t-1}||^2]$:

$$(2\eta-\eta^2)\mathbb{E}[||\boldsymbol{\theta}_t||^2] = \gamma^2C^2(1+2\frac{(1-\eta)(1-\alpha)}{\eta+\alpha-\alpha\eta}) \tag{3}$$

$$\mathbb{E}[||\boldsymbol{\theta}_t||^2] = \frac{\gamma^2C^2}{2\eta-\eta^2}\frac{(2-\eta-\alpha+\alpha\eta)}{\eta+\alpha-\alpha\eta} \tag{4}$$

Typically $\eta \ll \alpha \leq 1$. Ignore $O(\eta^2)$ and $O(\eta^3)$ terms of the denominator and $O(\eta)$ terms of the numerator, we get

$$\mathbb{E}[||\boldsymbol{\theta}_t||^2] \approx \gamma^2C^2\frac{2-\alpha}{2\alpha\eta} = \frac{\gamma_{\text{eff}}^2C^2}{2\eta} \tag{5}$$

$$= \gamma C^2\frac{2-\alpha}{2\alpha\lambda} \tag{6}$$

Eq. 5 again suggests that weight decay should be set $\propto \gamma^2$ and TUC of the minibatch is better characterized by the effective learning rate $\gamma_{\text{eff}} := \gamma\sqrt{\frac{2-\alpha}{\alpha}}$ at steady state. For Nesterov momentum, the effective learning rate is $\gamma_{\text{eff}} := \gamma\sqrt{\frac{2-\alpha}{\alpha}}(1 + 4\alpha - 6\alpha^2 + 2\alpha^3)^{-\frac{1}{2}}$ following similar derivation (Appx. A).

2.2.1 Input and hidden layers

We expect the analysis above to be applicable to input and hidden layers with L_2-norm-based lmos including Bias, RowNorm, and ColNorm in Pethick et al. (2025). For the Spectral lmo, the Frobenius norm of the update is proportional to $||\boldsymbol{U}\boldsymbol{V}^\top||_F = \sqrt{\min(d_{\text{in}}, d_{\text{out}})}$, so we postulate that its behavior at steady state is also similar to that of L_2 normalization. Numerical simulations suggest that this approximation is accurate for tall or wide matrices while overestimating the steady-state weight norm by $\approx 10\%$ for square matrices (Appx. B.2). One possible explanation for $\mathbb{E}[\langle \boldsymbol{u}_{t-1}, \boldsymbol{u}_t\rangle] = 0$ to hold for input and hidden layers at steady state is that the loss surface of deep learning models is multifractal (Ly & Gong, 2025), so there is little correlation between gradients at consequent steps on the loss surface.

2.2.2 Output layer

In the case of classification or next-token prediction models, the output layer is performing multinomial logistic regression (Cheng et al., 2018) so we do not expect $\cos(\boldsymbol{\theta}_{t-1}, \boldsymbol{g}_t) = \frac{\langle \boldsymbol{\theta}_{t-1}, \boldsymbol{g}_t\rangle}{||\boldsymbol{\theta}_{t-1}||||\boldsymbol{g}_t||}$ to vanish. Its expectation value determines the momentum-dependency of the steady-state weight norm, which is ironically harder to predict for the output layer. Qualitatively we expect $\cos(\boldsymbol{\theta}_{t-1}, \boldsymbol{u}_t)$ and therefore $||\boldsymbol{\theta}_t||$ to increase as α decreases since we are averaging $\boldsymbol{g}_t$ over more steps, and that setting weight decay $\propto \gamma$ results in stable weight norm since the contribution of the scalar projection of the update $u_{t||} = \cos(\boldsymbol{\theta}_{t-1}, \boldsymbol{u}_t)||\boldsymbol{u}_t||$ dominates. We therefore propose Scion with corrected weight decay (ScionC, Algorithm 2). Note that we schedule the effective learning rate $\gamma_{\text{eff},t,l}$ since it is

Algorithm 2 Scion with corrected weight decay (ScionC)

Input: Initial values $\boldsymbol{\theta}_{0,l}$, layer-wise effective learning rate schedule $\gamma_{\text{eff},t,l}$, choice of lmo_l for all layers l, Momentum schedule α_t, whether momentum is Nesterov, steady-state norm squared C_l^2 for input / hidden layers and weight decay coefficient λ_l for output layer
for layer $l = 0$ **to** L **do**
  $\boldsymbol{m}_{0,l} = 0$
end for
for $t = 1$ **to** T **do**
  for layer $l = 0$ **to** L **do**
    $\boldsymbol{g}_{t,l} = \nabla_{\theta_l} f_t(\boldsymbol{\theta}_{t-1,l}, \boldsymbol{\zeta}_t)$ ▷ Minibatch gradient
    $\boldsymbol{m}_{t,l} = (1 - \alpha_t)\boldsymbol{m}_{t-1,l} + \alpha_t \boldsymbol{g}_{t,l}$
    if Nesterov **then**
      $\boldsymbol{m}_{t,l} = (1 - \alpha_t)\boldsymbol{m}_{t,l} + \alpha_t \boldsymbol{g}_{t,l}$
    end if
    if l is an input / hidden layer **then**
      $\gamma_{t,l} = \gamma_{\text{eff},t,l}\sqrt{\frac{\alpha_t}{2-\alpha_t}}(1 + 4\alpha_t - 6\alpha_t^2 + 2\alpha_t^3)^{\frac{1}{2}}$ **if** Nesterov **else** $\gamma_{\text{eff},t,l}\sqrt{\frac{\alpha_t}{2-\alpha_t}}$
      $\eta_{t,l} = \frac{\gamma_{\text{eff},t,l}^2}{2C_l^2}$
    else
      $\gamma_{t,l} = \gamma_{\text{eff},t,l}$
      $\eta_{t,l} = \gamma_{t,l}\lambda_l$
    end if
    $\boldsymbol{\theta}_{t,l} = (1 - \eta_{t,l})\boldsymbol{\theta}_{t-1,l} + \gamma_{t,l}\,\text{lmo}_l(\boldsymbol{m}_{t,l})$
  end for
end for

what is proportional to the TUC of the minibatch. We then back out the learning rate $\gamma_{t,l}$ to handle the case of momentum scheduling.

2.3 AGGREGATED NEAR-SCALE-INVARIANCE

Some literature has proposed the idea that it is $\frac{||\boldsymbol{u}_t||}{||\boldsymbol{\theta}_{t-1}||}$ that determines training dynamics, not the update magnitude $||\boldsymbol{u}_t||$ by itself due to the scale-invariance introduced by the normalization layer (Kosson et al., 2024; Wen et al., 2026). Not all layer outputs are normalized in common architectures such as pre-norm transformers, however, so they do not exhibit total scale-invariance (Liu et al., 2026) and we need a weaker yet verifiable condition. One of the options is Hyp. 1:

Hypothesis 1: Aggregated Near-Scale-Invariance

Given the same training data distribution and the same model architecture with parameters $\boldsymbol{w}$ in the input and hidden layers and their output embedding $\boldsymbol{o}$ before the output layer, at the steady state:

$$\overline{\|\mathbf{J_o}\|_{\mathrm{F}}} \approx \frac{c}{\|\boldsymbol{w}\|_2}$$

where $\mathbf{J_o}$ is the Jacobian matrix of the output embedding w.r.t. the model parameters, $\overline{\|\mathbf{J_o}\|_{\mathrm{F}}}$ is its mean Frobenius norm, and $\|\boldsymbol{w}\|_2$ is the L_2 weight norm of the input and hidden layers.

Conceptually, $||\boldsymbol{w}||$ determines the input scale of the loss landscape. The update norm $\gamma||\boldsymbol{u}_t||$ therefore needs to be proportional to it to result in the same training dynamics. Assume Hyp. 1, one would expect the training dynamics of the input and hidden layers to be determined by the schedule of the relative learning rate $\gamma_{\mathrm{rel}} :- \frac{\gamma_{\mathrm{eff}}}{C}$ instead of the effective learning rate γ_{eff} alone for ScionC (Algorithm 2). We test this hypothesis and its implications in the following experiments.

3 EXPERIMENTS

3.1 SETUP

We test the idea of effective learning rate and aggregated near-scale-invariance by training a ViT-S/16 on the ImageNet-1k dataset (Russakovsky et al., 2015) and a 124M Modded-NanoGPT on FineWeb-Edu-100BT-shuffled (Niklaus & Kydlíček, 2026; Penedo et al., 2024) with ScionC and compare the result to that of the Scion baseline. For the ViT experiments, we base our setup on Beyer et al. (2022) and the changes made by Pethick et al. (2025) for optimizing ViT on Scion. For the Modded-NanoGPT experiments, we follow the setup of Pethick et al. (2025) except the use of fused linear cross entropy and batch size 1024×1024 (seqlen). We run automated hyperparameter tuning on 90 epochs of ImageNet-1k training set for the ViT and $3/10$ of FineWeb-Edu-100BT-shuffle for the Modded-NanoGPT. For the learning rate schedule when applicable, we either graft a known optimal schedule or run nested schedule tuning.

3.2 EFFECTIVE AND RELATIVE LEARNING RATES OF SCIONC

Fig. 2 and Appx. C show the metrics of the models with various momentum α vs. the input / hidden layer learning rate and effective learning rate. We can see that as we vary the momentum 50-fold, the optimal effective learning rate transfers within a factor of $\sqrt{2}$ while the optimal learning rate varies by almost a factor of 10. The input + hidden layer weight norm is approximately constant until the approximation breaks down around $\alpha = 0.2$–0.5 while the output weight norm increases as α decreases as expected. The models' training dynamics is even more invariant along the equi-γ_{rel} lines of $\gamma_{\mathrm{eff}} \propto C$ (Fig. 3). By backpropagating the dot product between the output embeddings and a random sign (Rademacher) vector, we calculate $\overline{\|\mathbf{J_o}\|_{\mathrm{F}}}$ of the models on the validation set with Hutchinson's trace estimator (Hutchinson, 1990) and confirm that $\overline{\|\mathbf{J_o}\|_{\mathrm{F}}} \approx \frac{c}{\|\boldsymbol{w}\|_2}$ (Fig. 4), in support of Hyp. 1.

3.3 BASELINE COMPARISON

We compare ScionC to the Scion baseline for two variants of simple ViT-S/16: one with bias on the hidden layers and one without. For the former we finish tuning the Scion baseline first and observe

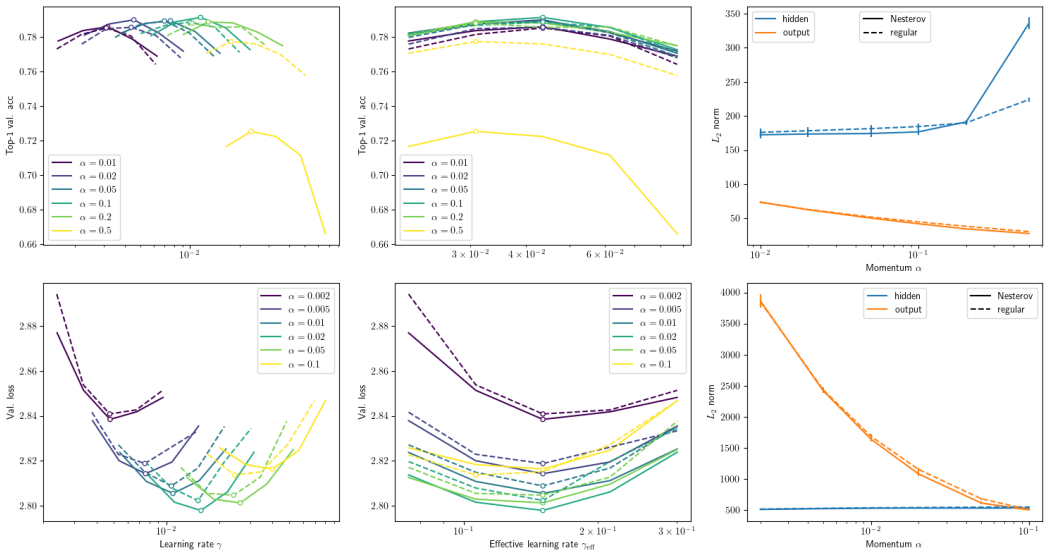


Figure 2: Top-1 val. accuracy of the ViT-S/16 after training for 90 epochs (top) and final val. loss of the 124M Modded-NanoGPT after training on $3/10$ of FineWeb-Edu-100BT-shuffle (bottom) with various Nesterov (solid) and regular (dashed) momentum α vs. the learning rate (left) and effective learning rate (middle) and the final weight norm of the input + hidden layers and output layer (right).

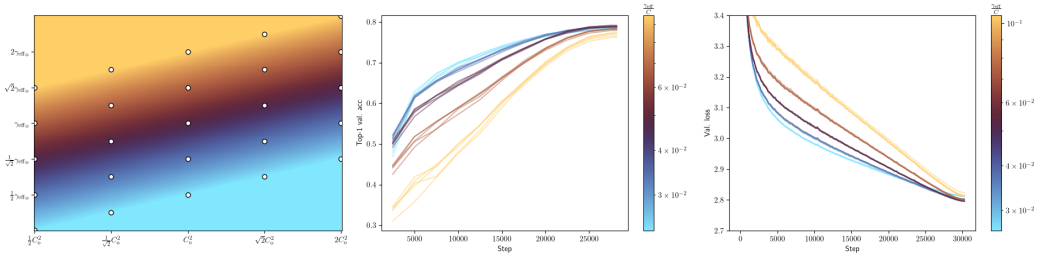


Figure 3: Sampled effective learning rate γ_{eff} and C^2 in terms of their optima $\gamma_{\text{eff}_\text{o}}$ and C_o^2 (left), the top-1 val. accuracy of the ViT-S/16 (middle), and the val. loss of the 124M Modded-NanoGPT (right) throughout the training process, color-coded by the corresponding relative learning rate $\gamma_{\text{rel}} := \frac{\gamma_{\text{eff}}}{C}$.

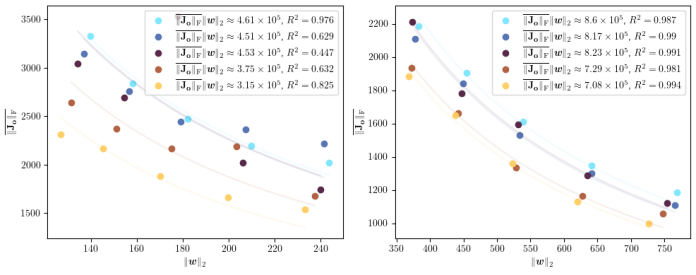


Figure 4: Estimated $\overline{\|\mathbf{J_o}\|_\text{F}}$ vs. the final input + hidden weight norm $\|\boldsymbol{w}\|_2$ of the ViT-S/16 (left) and Modded-NanoGPT (right) of Fig. 3 and R^2 of linear regression between $\overline{\|\mathbf{J_o}\|_\text{F}}$ and $\frac{1}{\|\boldsymbol{w}\|_2}$.

Table 2: ImageNet-1k top-1 val. accuracy ($N = 3$) of simple ViT-S/16 (left) and final validation loss of Modded-NanoGPT (right) trained with {ScionC, Scion} and various training budgets.

	with bias		without bias	
	ScionC	Scion	ScionC	Scion
30ep	**74.42**±0.07	73.05±0.18	**74.61**±0.14	74.12±0.12
60ep	**77.92**±0.14	77.68±0.09	**78.06**±0.08	77.99±0.15
90ep	79.00±0.16	**79.02**±0.17	79.14±0.06	**79.18**±0.17
150ep	79.52±0.03	**79.58**±0.03	79.69±0.06	**79.72**±0.12
300ep	79.50±0.18	**79.74**±0.27	79.75±0.07	**79.75**±0.06

	ScionC	Scion
$^3/_{10}$	**2.796**	2.800
full	**2.752**	2.760

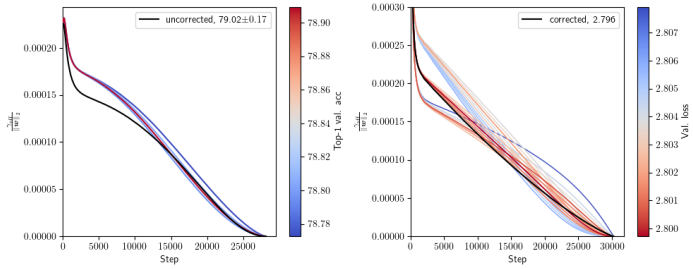


Figure 5: Simple ViT-S/16 LR schedule grafting for ScionC (left) and Modded-NanoGPT nested LR schedule tuning for Scion (right), color-coded by the respective performance metric. Since γ_{rel} is undefined for Scion, we plot γ_{eff} over (input +) hidden layer norm directly for both instead. ScionC with $\frac{\gamma_{\text{eff}}}{||\boldsymbol{w}||_2}$ schedule closer to that of Scion optimum performs better and vice versa.

that ScionC is highly sensitive to the γ_{rel} schedule, so we focus on testing γ_{rel} schedules similar to that of optimized Scion as a form of grafting (Fig. 5L). Interestingly, after grafting the 90ep LR schedule ScionC and the Scion baseline hold advantage for shorter and longer training budgets respectively, leaving the question of optimal γ_{rel} for all budgets open (Table 2L). The question is rendered somewhat moot by the variant without bias that performs better and is less sensitive to the LR schedule, so we perform the momentum and scale sweeps in Sec. 3.2 with the latter variant and cosine LR schedule for simplicity.

The situation for Modded-NanoGPT is the opposite: Due to the dependence of $\|\boldsymbol{w}\|_2$ on γ_{eff}, it is difficult to graft an optimized LR schedule for ScionC to the Scion baseline with a preset schedule. Nested schedule tuning suggests that the Scion baseline performs better the closer the LR schedule is to the ScionC optimum but can't fully match (Fig. 5R, Table 2R).

3.4 Log-time Momentum Scheduling

Finally, recent experiments suggest that momentum should be scheduled as $\alpha \propto \frac{\delta}{\delta+t}$ to match the frequency distribution of language features for LLM training (Ferbach et al., 2026) and knowledge of γ_{eff}'s dependence on momentum allows us to test log-time momentum while controlling for the γ_{eff} schedule with ScionC. While the optimal momentum α for training the 124M Modded-NanoGPT on $^3/_{10}$ of the FineWeb-Edu-100BT-shuffled is $\alpha = 0.02$ (Fig. 2), the optimal constant momentum for training on the full FineWeb-Edu-100BT-shuffled is $\alpha \leq 0.002$ and the optimal log-time momentum is $\alpha = 0.05 \to 0.00062$ (Appx. D, Fig. 6) in support of the hypothesis.

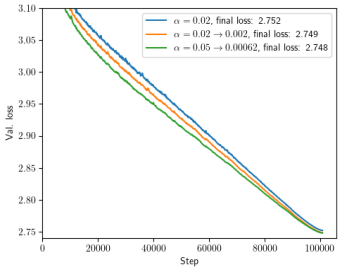


Figure 6: Val. loss of Modded-NanoGPT trained with the full FineWeb-Edu-100BT-shuffled and momentum schedules.

4 Related Works

Correction of decoupled weight decay. Kosson et al. (2024) derives the steady-state weight norm $\sqrt{\frac{\gamma C}{2\lambda}}$ and Defazio (2025) proposes AdamC which keeps the

weight norm stable by maintaining $\lambda \propto \gamma$ throughout the learning rate schedule. Apte (2026) has tested Muon with $\lambda \propto \gamma$ on Mixture-of-Experts (MoE, Shazeer et al. (2017)) transformer models. In addition, Inkling is also reported to be a MoE transformer model trained with Muon and corrected weight decay (Thinking Machines Lab, 2026).

Hyperball counterpart. Concurrent works (Wen et al., 2026; Liu et al., 2026; Zhou et al., 2026; Xiao et al., 2026) have compared the optimal LR schedule for hyperball optimizers to that of optimizers with conventional (uncorrected) weight decay and arrived at the same conclusion that it is $\frac{\gamma}{||\boldsymbol{w}||_2}$ that determines the training dynamics.[1] In particular, up to notation differences Wen et al. (2026) derives the equilibrium norm identical to Eq. 4 for L_2-norm-based lmos and nearly-identical loss trajectories reported in Fig. 3R are called "ELR collapse " in Liu et al. (2026). We derive momentum-dependent γ_{eff} based on the approximate Eq. 5 and focus on the case of corrected weight decay, whose soft control nonetheless results in sufficiently stable weight norm for γ_{rel} comparison and grafting.

Momentum scheduling. In addition to ADANA's log-time momentum (Ferbach et al., 2026), Defazio (2026) also reports that log-linear interpolation with time-varying β improves LLM training. Such momentum schedule is equivalent to exponential-decaying α that we test along with log-time momentum (Appx. D).

5 Conclusions

5.1 Implications

"RMS-matching Muon". Muon(Clip) used by Kimi (Kimi Team et al., 2026) and GLM (Team et al., 2025) has a LR RMS-matching factor of 0.2 in order to reuse AdamW's optimal LR. While the RMS of the AdamW update is indeed $\sqrt{\frac{1-\beta_1}{1+\beta_1}} \approx 0.23$ when $\beta_1 = 0.9$ (Kosson et al., 2024; Su, 2025), the optimal LR of AdamW does not depend on the value of β_1 (Orvieto & Gower (2025), Appx. E) so RMS matching does not explain the factor of 0.2. Instead, the factor of 0.2 is due to the effective LR of Muon. With $\mu = 0.95$ used by Kimi and GLM, $\gamma_{\text{eff}} = \gamma\sqrt{\frac{1+\mu}{1-\mu}} \approx 6.245\gamma$ (Appx. A.2), so a factor of 0.2 brings it back to $\approx 1.25\gamma$, approximately the raw learning rate. Similarly, DeepSeek opted to train DeepSeek V4 and V4.1 Flash (DeepSeek-AI et al., 2026b;a) with Nesterov Muon, $\mu = 0.95$, and a LR RMS-matching factor of 0.18 so $0.18\gamma_{\text{eff}} = 0.18\gamma\sqrt{\frac{1+\mu}{1-\mu}}(1 + 2\mu - 2\mu^3)^{-\frac{1}{2}} \approx 1.03\gamma$ is even closer to the raw learning rate γ.

Layer-wise $(\mathbf{L^0}, \mathbf{L^1})$-smoothness and optimal LR Riabinin et al. (2025) advocates a layer-wise (L^0, L^1)-smoothness model for the loss landscape of deep learning models and claims to derive the approximate optimal LR for Scion based on the estimated values of L^0 and L^1. Since the derivation does not take momentum-dependence into account, however, it is at best incomplete. In fact, our preliminary attempts to adapt the LR to running estimates of L^0 and L^1 rapidly diverge.

5.2 Recommendations

We recommend optimizer parameterizations that disentangle learning rate, weight norm control, and momentum like ScionC. Not because it would automatically perform better, but because it allows us to isolate the effect of and optimize each hyperparameter. In case there is already a performing baseline, we can graft its relative learning rate (learning rate over weight norm) as the new starting point (Fig. 5L). Relatedly, we recommend separate optimizer parameterizations for input + hidden layers and the output layer given how much their training dynamics differ. We follow the common practice and use the same momentum α for the whole model to gather data in Fig. 2, but in practice we recommend separate momenta and keep the output layer momentum constant to avoid weight norm swings, e.g. we fix $\alpha = 0.02$ for the output layer for the momentum scheduling experiments in Fig. 6 and Appx. D.

[1] It is more common to call $\frac{\gamma}{||\boldsymbol{w}||_2}$ "effective learning rate" (ELR) in the hyperball literature but we need to distinguish γ_{rel} from momentum-dependent γ_{eff} and ScionC does not directly specify $||\boldsymbol{w}||_2$.

AI USE STATEMENT

We brainstormed the derivation and approximation of the steady-state weight norm in the case of momentum with normalized update (Sec. 2.2, Appx. A) with DeepSeek R1 (Guo et al., 2025) and 3.2 (DeepSeek-AI et al., 2025). The writing and editing of the other sections involved no AI assistance and neither did the implementation, scripting, analysis, and plotting of the experiments. We have reviewed all AI-assisted work and fully understand the results. We take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

OPTIMIZERS AND MODELS

Our Scion baseline generally follows the official implementation of Pethick et al. (2025) except for

1. The reparameterization described in Sec. 2.1
2. Improvement in efficiency through sharding the state variables and parameter updates on multi-GPU nodes in the spirit of Rajbhandari et al. (2020)
3. Improved reduced SVD accuracy with PolarExpress (Amsel et al., 2025).

We then further modify the multi-GPU Scion to implement ScionC. For the ViT experiments, we follow the model architecture (sometimes called "Simple ViT") and setup of Beyer et al. (2022) including sincos2d positional encoding, global average pooling (GAP), and data augmentations including RandAugment (Cubuk et al., 2020) and Mixup (Zhang et al., 2018). The only exceptions are the PyTorch Inception crop with crop scale lower bound $a_{min} = 0.05$ and the changes made by Pethick et al. (2025) for DeiT (Touvron et al., 2020):

1. Scale the GELU activation function as $\sqrt{2}$GELU to preserve variance
2. Replace LayerNorm with RMSNorm
3. Batch size 4096

For the Modded-NanoGPT experiments, we follow the setup of Pethick et al. (2025) except the fused linear cross entropy of PyTorch 2.13, which in turn allows us to run batch size 1024×1024 (seqlen) instead of 512×1024 on $8\times$H100.

HYPERPARAMETER TUNING

We run automated hyperparameter tuning on 90 epochs of ImageNet-1k training set for the ViT and $3/10$ of FineWeb-Edu-100BT-shuffle for the Modded-NanoGPT after leaving out $10\times1024\times1024+1$ tokens as validation set. The scripts tune input + hidden layer LR and WD (C^2 for ScionC and λ for Scion), Nesterov (or regular momentum), momentum α, output (Sign) layer LR and WD in that order (estimated descending order of importance). The scripts perform simple hill climbing, maximizing the best top-1 val. accuracy for ViT-S/16 and minimizing the final val. loss for the Modded-NanoGPT. We run $N = 1$ for Modded-NanoGPT and $N = 3$ for ViT-S/16 if the difference in top-1 val. accuracy is $< 0.2\%$[2] between the first runs. If the difference is $\geq 0.2\%$ between the first runs, the scripts skip repeated runs and hill-climbs in the implied direction. Nesterov or regular momentum is a boolean flag and we use momentum values of the form $\alpha = \{1, 2, 5\} \times 10^{-n}, n \in \mathbb{N}$ which are more readable and common in the optimizer literature. The granularity of all the other hyperparameters is fixed to be factors of $\sqrt{2}$. Most of the time this sequential tuning finds at least a local optimum except that when momentum α is too high regular momentum is preferred over Nesterov momentum (Fig. 2), forcing us to retune ScionC with the optimal Nesterov momentum for Modded-NanoGPT. Since Scion experiments start with the hyperparameters tuned in the first pass of ScionC, such retuning only puts ScionC on an equal footing.

We deploy nested tuning for two cases: momentum scheduling (Appx. D) and LR schedule tuning. For LR schedule, we consider two functional forms: fractional polynomial $\gamma_t = \gamma(1 - \frac{t}{T})^p$ and generalized cosine (Naganuma et al., 2026) $\gamma_t = \gamma(\frac{1+\cos(\frac{\pi t}{T})}{2})^p$. For each functional form, we

[2] Approximately the standard deviation of repeated runs with the same settings (Table 2L).

find the optimal power p for each LR value $(\ldots, \frac{1}{\sqrt{2}}\gamma, \gamma, \sqrt{2}\gamma, \ldots)$ until we find a local optimum (γ_o, p_o) that also outperforms (γ, p) $\forall\gamma \in \{\frac{1}{\sqrt{2}}\gamma_o, \sqrt{2}\gamma_o\}\forall p$. In practice we find it difficult for a single granularity value for power p to find a good optimum within a reasonable budget, so eventually we adapt coarse-to-fine hill climbing with coarse granularity $\Delta p = 0.03$ for the ViT and $\Delta p = 0.02$ for the Modded-NanoGPT and fine granularity $\Delta p = 0.01$. In the end we pick the better of the best fractional polynomial schedule and the best generalized cosine schedule.

Repositories and Scripts

We use one repository for the ViT-S/16 experiments and another for the Modded-NanoGPT experiments. We perform experiments with pairs of scripts consist of Python script generator and its Bash script driver:

ViT-S/16 Experiments

1. Variant without hidden layer bias, ScionC and Scion (py, sh)
2. Variant with hidden layer bias, ScionC (py, sh)
3. Variant with hidden layer bias, Scion (py, sh)

Modded-NanoGPT Experiments

1. First pass of ScionC hyperparameter tuning and Scion experiments (py, sh)
2. ScionC retuning, momentum scheduling, and RowNorm experiments (py, sh)

We will provide better documented and more complete source code after the review process.

References

Noah Amsel, David Persson, Christopher Musco, and Robert M. Gower. The polar express: Optimal matrix sign methods and their application to the muon algorithm, 2025. URL https://arxiv.org/abs/2505.16932.

Anuj Apte. Scale weight decay and train better, 2026. URL https://arxiv.org/abs/2607.23777.

Lucas Beyer, Xiaohua Zhai, and Alexander Kolesnikov. Better plain vit baselines for imagenet-1k, 2022. URL https://arxiv.org/abs/2205.01580.

Bowen Cheng, Rong Xiao, Yandong Guo, Yuxiao Hu, Jianfeng Wang, and Lei Zhang. Revisit multinomial logistic regression in deep learning: Data dependent model initialization for image recognition. *arXiv preprint arXiv:1809.06131*, 2018.

Ekin Dogus Cubuk, Barret Zoph, Jon Shlens, and Quoc Le. Randaugment: Practical automated data augmentation with a reduced search space. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 18613–18624. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/d85b63ef0ccb114d0a3bb7b7d808028f-Paper.pdf.

DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020. URL http://github.com/google-deepmind.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Erhang Li, Fangqi Zhou, Fangyun Lin, Fucong Dai, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoran Wei, Haowei Zhang, Haowen Luo, Haozhe Ji, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, Jialiang Huang, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jingchang Chen, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jinhua Zhu, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexin Huang, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Liang Zhao, Liangsheng Yin, Lihua Guo, Lingxiao Luo, Linwang Ma, Litong Wang, Liyue Zhang, M. S. Di, M. Y Xu, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Panpan Huang, Peixin Cong, Peiyi Wang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, S. H. Liu, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Songyang Zhou, Tao Ni, Tao Yun, Tian Pei, Tian Ye, Tianyuan Yue, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjun Gao, Wentao Zhang, Xi Gao, Xiangwen Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyuan Li, Xu Chen, Xuecheng Su, Xuehai Pan, Xuheng Lin, Xuwei Fu, Y. Q. Wang, Yang Zhang, Yanhong Xu, Yanru Ma, Yao Li, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yiliang Xiong, Ying He, Ying Zhou, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuduan Wang, Yue Gong, Yuhan Wu, Yuheng Zou, Yukun Li, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehua Zhao, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhiyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Zizheng Pan, Zongqing Yao, Bei Feng, Hui Li, J. L. Cai, Jiaqi Ni, Lei Xu, Meng Li, Ning Tian, R. J. Chen, R. L. Jin, S. S. Li, Shuang Zhou, Tianyu Sun, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xinnan Song, Xinyi Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, Dongjie Ji, Jian Liang, Jianzhong Guo, Jin Chen, Leyi Xia, Miaojun Wang, Mingming Li, Peng Zhang, Ruyi Chen, Shangmian Sun, Shaoqing Wu, Shengfeng Ye, T. Wang, W. L. Xiao, Wei An, Xianzu Wang, Xiaowen Sun, Xiaoxiang Wang, Ying Tang, Yukun Zha, Zekai Zhang, Zhe Ju, Zhen Zhang, and Zihua Qu. Deepseek-v3.2: Pushing the frontier of open large language models, 2025. URL https://arxiv.org/abs/2512.02556.

DeepSeek-AI, :, Anyi Xu, B. Li, Bangcai Lin, Bing Xue, BingCheng Xian, Bingzheng Xu, Bochao Wu, Bowei Zhang, Boyi Deng, C. C. Yu, Chao Jin, Chaofan Lin, Chen Dong, Chenbing Wang, Chenfan Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chengyuan Zhang, Chenhao Xu, Chenqi Zhao, Chenze Shao, Chuhao Wang, Chuqi Zhang, Damai Dai, Dejian Yang, Deli Chen, Di Huang, Di Wu, Donghao Li, Erhang Li, Eric Fu, F. Zhou, Fangwei Zhou, Fangyun Lin, Fangzhou Yuan, Feiyu Xia, Fucong Dai, Guangbo Hao, Guanglin Li, Guanting Chen, Guoai Cao, Guofan Fan, Guolai Meng, Guowei Li, Haichuan Zhang, Haiyang Ma, Haiyang Shen, Han Li, Han Yu, Han Zhang, Hangyuan Deng, Hanwei Xu, Hanxiang Xu, Hanxun Zhong, Hao Guo, Hao Jiang, Hao Li, Hao Qin, Haodong Wen, Haofen Liang, Haofeng Huang, Haohua Liu, Haoling Zhang, Haoming Luo, Haoran Yang, Haotian Xu, Haotian Yuan, Haoting Huang, Haowen Luo, Haoyang Cai, Haoyu Chen, Haozhe Ji, Hengran Zhang, Hengrui Wang, Hengxu Wu, Honghui Ding, Hongxuan Tang, Huadong Wang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, J. Yang, J. H. Jin, J. H. Zhang, J. X. Zou, Jia Yu, Jiahui Zhou, Jiajun Chen, Jialiang Huang, Jialin Zhao, Jiamin Tang, Jian Zhou, Jianan Tong, Jianwen Li, Jiaqi Zhu, Jiarui Wang, Jiasheng Ye, Jiashi Li, Jiaxin Xu, Jiaying Ding, Jibai Lu, Jiewen Hu, Jin Yan, Jincheng Zhai, Jingchang Chen, Jingcheng Hu, Jingli Zhou, Jingsheng Xu, Jingting Xiang, Jingyan Yun, Jingyang Yuan, Jingyuan Cheng, Jinhua Zhu, Jinpeng Wang, Jinyi Chen, Jinyi Hu, Jiping Yu, Jueliang Guo, Junbo Pei, Junbo Sun, Junguang Jiang, Junjie Qiu, Junkang Zhou, Junqi Liu, Junren Li, Junxian Li, Junxiao Song, Junyi Guo, Kai Dong, Kaifeng Chen, Kaige Gao, Kang Guan, Kangdong Yuan, Ke Hong, Ke Xu, Kefan Zhao, Kexin Ji, Kexin Zhang, Kexing Zhou, Kuai Yu, Lan Zhang, Lean Wang, Lecong

Zhang, Lei Wang, Letian Gao, Liang Zhao, Liansheng Xu, Lihua Guo, Lingxiao Luo, Lingyue Fu, Litao Deng, Litong Wang, Liyue Zhang, Longhao Chen, Lu Chen, Luotian Huang, Luyao Ma, Luyao Wang, M. S. Di, Max Mei, Menghao Ye, Miao Cui, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingjing Zhang, Mingqi Wei, Mingshu Chen, Mingxing Liu, Mingxu Zhou, Mingyu Xu, Mingyu Yang, Mingze Wang, Muyang Chen, Ni Shentu, Ning Wang, Niufang Ning, Panpan Huang, Peixin Cong, Peiyi Wang, Peiyuan Xin, Pengfei Ren, Pengfei Yan, Pengle Zhang, Qi Kang, Qi Tang, Qiancheng Wang, Qiang Li, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Qizhou Guo, Rongxian Xu, Rui Ding, Rui Hu, Rui Tian, Rui Yu, Ruidong Zhu, Ruifan Xu, Ruihan Yang, Ruihang Xia, Ruijie Lu, Ruilin Geng, Ruipeng Hong, Ruiqi Ge, Ruisong Zhang, Ruize Sun, Ruizhe Pan, Runji Wang, Runqian Chen, Runxin Xu, Ruohong Tian, Ruomeng Shen, Ruoyu Zhang, Ryan X., S. H. Liu, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoheng Nie, Shaoyuan Chen, Shengding Hu, Shengkai Lin, Shengwen Ran, Shengyu Liu, Shengyuan Jia, Shi Bai, Shi Feng, Shicheng Xu, Shichun Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shiyuan Feng, Shufan Gong, Shuhan Lin, Shuiping Yu, Shunfeng Zhou, Shuo Yang, Shuomeng Wang, Shuting Guo, Shuting Pan, Shuying Yu, Sinuo Cao, Siyi Lin, Sizhe Chen, Songyang Chen, Songyang Zhou, Tao Ni, Tao Yun, Tian Jin, Tian Pei, Tian Ye, Tianle Lin, Tianran Ji, Tianyi Cui, Tianyuan Yue, Tingting Yu, Tongrui Xiong, Wangding Zeng, Wei Liu, Wei Zhang, Weibin Xu, Weihao Zeng, Weilin Zhao, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjing Yao, Wenjun Gao, Wenkai Shao, Wenkai Yang, Wenli Zhang, Wenlu Wang, Wenlve Huang, Wenqian Yan, Wentao Zhang, Xi Gao, Xiang He, Xiang Li, Xiangli Li, Xiangwen Wang, Xiangying Zhang, Xiankui Wei, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojian Qu, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xiaoyao Zou, Xiaoyuan Li, Xicheng Guo, Xieting Chu, Xin Cheng, Xin Liu, Xin Xie, Xinbo Xu, Xingchao Liu, Xingchen Liu, Xingkai Yu, Xingyou Li, Xintong Yao, Xinyang Chen, Xinyong Jiang, Xinyu Yang, Xinyu Yang, Xu Chen, Xuanyu Wang, Xubei Zhong, Xuecheng Su, Xuejie Liu, Xuheng Lin, Xujie Fan, Xuncheng Zhao, Xuwei Fu, Y. C. Yan, Y. H. Jiang, Y. T. Wu, Y. W. M., Y. Z. Wang, Yafei Gao, Yang Yang, Yang Zhang, Yanru Ma, Yanwen Huang, Yao Li, Yao Li, Yao Meng, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yaoyang Ye, Yehang Yin, Yexinrui Wu, Yi Qian, Yi Tao, Yi Yu, Yichao Zhang, Yichen Jiang, Yicheng Wang, Yifan Ding, Yifan Shi, Yifeng Peng, Yifeng Zhai, Yijia Wu, Yiliang Xiong, Yilun Wang, Ying He, Ying Zhou, Yingjia Luo, Yinmin Zhong, Yiping Wang, Yisong Wang, Yixiang Zhang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyao Yang, Yiyuan Liu, Yizai Cai, Yizhen Wei, Yizhi Wang, Yonglun Yang, Yongqi Zhuo, Yongqiang Guo, Yongtong Wu, Yu Wu, Yu Zhang, Yuan Bian, Yuan Cheng, Yuan Ou, Yuan Sun, Yuanfan Xu, Yuanhang Sun, Yuanhao Li, Yuchen Liu, Yuchen Yao, Yudong Han, Yuduan Wang, Yuhan Wu, Yuhao Meng, Yuheng Zou, YuKun Li, Yunchuan Wang, Yunfan Xiao, Yunfan Xiong, Yupeng Chen, Yuqian Cao, Yuqian Wang, Yuqing Chen, Yushun Zhang, Yutong Lin, Yuwei Xiao, Yuxian Gu, Yuxiang Chen, Yuxiang Huang, Yuxiang Luo, Yuxiang You, Yuxin Chen, Yuxin Xiang, Yuxuan Liu, Yuxuan Zhou, Yuyang Zhou, Yuzhe Guo, Yuzhen Huang, Yuzhuo Bai, Z. Y. Z., Zanlin Ni, Zehao Wang, Zehua Zhao, Zehui Ren, Zejun Zhao, Zhangli Sha, Zhanying Wang, Zhaochen Zhang, Zhaoshuai Du, Zhe Fu, Zhean Xu, Zhenda Xie, Zheng Liu, Zhengyan Zhang, Zhenhua Dong, Zhewen Hao, Zhibang Wang, Zhibin Gou, Zhicheng Ma, Zhihao Li, Zhihong Shao, Zhihuan Huang, Zhijie Li, Zhirui Lu, Zhixian Huang, Zhixuan Chen, Zhixuan Chen, Zhixuan Pan, Zhiyu Wu, Zhizhou Ren, Zhu He, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihui Gu, Zijia Zhu, Zili Zhang, Zilin Li, Zilong Hou, Zilong Lyu, Ziqiao Wang, Ziwei Xie, Ziya Zhang, Ziyi Gao, Zizheng Pan, Zonglin Li, Zongqing Yao, Zui Chen, Zuofan Wu, Chenchen Ling, Chengyu Hou, Chong Chen, D. Li, Di Qi, Dongjie Ji, Fang Wei, Fanyi Xia, Fei Xie, Feiyi Tan, Hailong Guo, Haiyan Zhai, Hui Zhou, Huihui Tan, Huijie Li, Jia Luo, Jia Song, Jialu Cai, Jian Liang, Jiangting Zhou, Jiaqi Gao, Jiayi Shao, Jie Chen, Jieyu Yang, Jin Chen, Jingde Zhang, Jingzi Zhou, Jinqian Wang, Jinyang Liu, JinZhao Sun, Junhua Ling, Junmin Zheng, Kaicheng Yang, Ke Xu, Le Su, Leyi Xia, Liangfeng Ding, Lin Zhuo, Linwang Ma, Linyan Zhu, Liyu Cai, Luqi Yao, M. K. Zhang, Meng Li, Miao Lin, Miaojun Wang, Min Zhang, Mingming Li, Mingming Wang, Mingze Yin, Minmin Han, Nan Cao, Ning Wang, Ningxin Ma, Panpan Wang, Peihan Lin, Peng Sun, Peng Zhang, Qian Ying, Qiang Xiang, Qiao Wang, Qingmiao Mao, Qiwei Jiang, Rongli Jin, Ruyi Chen, Sha Tao, Shangmian Sun, Shaoqing Wu, Shichao Zou, Si Lei, Tianyang Zhang, Tianyu Sun, Tingting Yin, W. L. Xiao, Wei An, Wei Li, Wei Wang, Weiwei Lin, Wenqing Hou, X. Lin, Xiangfei Meng, Xianzhu Huang, Xiao Peng, Xiaoqian Li, Xiaoting Zhang, Xiaowen Sun, Xiaoxiang Wang, Xiaoyu Ye, Xinrou Zhang, Xinyu Zhang, Xue Cao, Xueyin Chen, Yanan Zhou, Yanhong Xu, Yao Xia, Yao Xu, Yi Shao, Yihong Zhang, Yiling Ma, Ying Tang, Yining Lou, Yiru Chen, Yishi Piao, Yixuan Chen, Yong Xiong, Yuchen Xuan, Yuehan Yang, Yuer Xu, Yukun Zha, Yunxian Ma, Yup-

ing Lin, Yuting Yan, Yutong Xie, Yuwen Sheng, Yuxuan Zhu, Zekai Zhang, Zhe Ju, Zhenzhen Lin, Zheren Gao, Zheyang Sun, Zhigang Yan, Zhongyu Wu, Zi Wang, Zihua Qu, Ziling Yan, and Ziyi Wan. Deepseek-v4.1-flash: Pushing the limits of kv cache compression, 2026a. URL https://arxiv.org/abs/2609.19969.

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chengyu Hou, Chenhao Xu, Chenze Shao, Chong Ruan, Conner Sun, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Donghao Li, Dongjie Ji, Erhang Li, Fang Wei, Fangyun Lin, Fangzhou Yuan, Feiyu Xia, Fucong Dai, Guangbo Hao, Guanting Chen, Guoai Cao, Guolai Meng, Guowei Li, Han Yu, Han Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoling Zhang, Haoming Luo, Haoran Wei, Haotian Yuan, Haowei Zhang, Haowen Luo, Haoyu Chen, Haozhe Ji, Hengqing Zhang, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, J Yang, JQ Zhu, Jia Luo, Jia Song, Jia Yu, Jialiang Huang, Jialu Cai, Jian Liang, Jiangting Zhou, Jiasheng Ye, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jieyu Yang, Jin Chen, Jin Yan, Jingchang Chen, Jingli Zhou, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jingzi Zhou, Jinhua Zhu, Jiping Yu, Joseph Sun, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junmin Zheng, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Leyi Xia, Li Zhang, Liang Zhao, Lihua Guo, Lingxiao Luo, Linwang Ma, Linyan Zhu, Litong Wang, Liyu Cai, Liyue Zhang, Longhao Chen, MS Di, MY Xu, Max Mei, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Mingxu Zhou, Minmin Han, Ning Wang, Panpan Huang, Panpan Wang, Peixin Cong, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Qiwei Jiang, Rui Tian, Ruifan Xu, Ruijie Lu, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqian Chen, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, Ruyi Chen, SH Liu, Shanghao Lu, Shangmian Sun, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoheng Nie, Shaoqing Wu, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Shuying Yu, Songyang Zhou, Tao Ni, Tao Yun, Tian Jin, Tian Pei, Tian Ye, Tianle Lin, Tianran Ji, Tianyi Cui, Tianyuan Yue, Tingting Yu, Tun Wang, W Zhang, WL Xiao, Wangding Zeng, Wei An, Weilin Zhao, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjing Yao, Wenjun Gao, Wenkai Yang, Wenlve Huang, Wenqing Hou, Wentao Zhang, Wenting Ma, Xi Gao, Xiang He, Xiangwen Wang, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingchen Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyu Zhang, Xu Chen, Xuanyu Wang, Xuecheng Su, Xueyin Chen, Xuheng Lin, Xuwei Fu, YC Yan, YQ Wang, YW Ma, Yanfeng Luo, Yang Zhang, Yanhong Xu, Yanru Ma, Yanwen Huang, Yao Li, Yao Li, Yao Xu, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Shao, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yijia Wu, Yiliang Xiong, Yiling Ma, Ying He, Ying Tang, Ying Zhou, Yingjia Luo, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiang Zhang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, YuKun Li, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuanhao Li, Yuduan Wang, Yuehan Yang, Yuer Xu, Yuhan Wu, Yuhao Meng, Yuheng Zou, Yukun Zha, Yunfan Xiong, Yupeng Chen, Yuping Lin, Yuqian Cao, Yuqian Wang, Yushun Zhang, Yuting Yan, Yutong Lin, Yuxian Gu, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuxuan Zhou, Yuyang Zhou, Yuzhen Huang, ZF Wu, Zehao Wang, Zehua Zhao, Zehui Ren, Zekai Zhang, Zhangli Sha, Zhe Fu, Zhe Ju, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zheren Gao, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhixuan Chen, Zhiyu Wu, Zhizhou Ren, Zhongyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihua Qu, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Ziyi Wan, Zizheng Pan, and Zongqing Yao. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026b. URL https://arxiv.org/abs/2606.19348.

Aaron Defazio. Why gradients rapidly increase near the end of training, 2025. URL https://arxiv.org/abs/2506.02285.

Aaron Defazio. Schedulefree+: Scaling learning-rate-free & schedule-free learning to large language models, 2026. URL https://arxiv.org/abs/2605.19095.

Damien Ferbach, Courtney Paquette, Gauthier Gidel, Katie Everett, and Elliot Paquette. Logarithmic-time schedules for scaling language models with momentum. *arXiv preprint arXiv:2602.05298*, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://doi.org/10.1038/s41586-025-09422-z.

M.F. Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics - Simulation and Computation*, 19(2):433–450, 1990. doi: 10.1080/03610919008812866. URL https://doi.org/10.1080/03610919008812866.

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

Kimi Team, Yifan Bai, Yiping Bao, Y. Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, Yichen Feng, Kelin Fu, Bofei Gao, Chenxiao Gao, Hongcheng Gao, Peizhong Gao, Tong Gao, Yuyao Ge, Shangyi Geng, Qizheng Gu, Xinran Gu, Longyu Guan, Haiqing Guo, Jianhang Guo, Xiaoru Hao, Tianhong He, Weiran He, Wenyang He, Yunjia He, Chao Hong, Hao Hu, Yangyang Hu, Zhenxing Hu, Weixiao Huang, Zhiqi Huang, Zihao Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yongsheng Kang, Guokun Lai, Cheng Li, Fang Li, Haoyang Li, Ming Li, Wentao Li, Yang Li, Yanhao Li, Yiwei Li, Zhaowei Li, Zheming Li, Hongzhan Lin, Xiaohan Lin, Zongyu Lin, Chengyin Liu, Chenyu Liu, Hongzhang Liu, Jingyuan Liu, Junqi Liu, Liang Liu, Shaowei Liu, T. Y. Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yibo Liu, Yiping Liu, Yue Liu, Zhengying Liu, Enzhe Lu, Haoyu Lu, Lijun Lu, Yashuo Luo, Shengling Ma, Xinyu Ma, Yingwei Ma, Shaoguang Mao, Jie Mei, Xin Men, Yibo Miao, Siyuan Pan, Yebo Peng, Ruoyu Qin, Zeyu Qin, Bowen Qu, Zeyu Shang, Lidong Shi, Shengyuan Shi, Feifan Song, Jianlin Su, Zhengyuan Su, Lin Sui, Xinjie Sun, Flood Sung, Yunpeng Tai, Heyi Tang, Jiawen Tao, Qifeng Teng, Chaoran Tian, Chensi Wang, Dinglu Wang, Feng Wang, Hailong Wang, Haiming Wang, Jianzhou Wang, Jiaxing Wang, Jinhong Wang, Shengjie Wang, Shuyi Wang, Si Wang, Xinyuan Wang, Yao Wang, Yejie Wang, Yiqin Wang, Yuxin Wang, Yuzhi Wang, Zhaoji Wang, Zhengtao Wang, Zhengtao Wang, Zhexu Wang, Chu Wei, Qianqian Wei, Haoning Wu, Wenhao Wu, Xingzhe Wu, Yuxin Wu, Chenjun Xiao, Jin Xie, Xiaotong Xie, Weimin Xiong, Boyu Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ziyao Xu, Jing Xu, Jing Xu, Junjie Yan,

Yuzi Yan, Hao Yang, Xiaofei Yang, Yi Yang, Ying Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Xingcheng Yao, Wenjie Ye, Zhuorui Ye, Bohong Yin, Longhui Yu, Enming Yuan, Hongbang Yuan, Mengjie Yuan, Siyu Yuan, Haobing Zhan, Dehao Zhang, Hao Zhang, Wanlu Zhang, Xiaobin Zhang, Yadong Zhang, Yangkun Zhang, Yichi Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Haotian Zhao, Yikai Zhao, Zijia Zhao, Huabin Zheng, Shaojie Zheng, Longguang Zhong, Jianren Zhou, Xinyu Zhou, Zaida Zhou, Jinguo Zhu, Zhen Zhu, Weiyu Zhuang, and Xinxing Zu. Kimi k2: Open agentic intelligence, 2026. URL https://arxiv.org/abs/2507.20534.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *ICLR (Poster)*, 2015. URL http://dblp.uni-trier.de/db/conf/iclr/iclr2015.html#KingmaB14.

Atli Kosson, Bettina Messmer, and Martin Jaggi. Rotational equilibrium: how weight decay balances learning across neural networks. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Zihan Liu, Ruiheng Zheng, Shaobo Zhang, Changxin Tian, Kunlong Chen, Zhiqiang Zhang, and Lei Wu. Effective learning rate governs loss dynamics in language model pretraining, 2026. URL https://arxiv.org/abs/2608.24814.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Andrew Ly and Pulin Gong. Optimization on multifractal loss landscapes explains a diverse range of geometrical and dynamical properties of deep learning. *Nature Communications*, 16 (1):3252, 2025. doi: 10.1038/s41467-025-58532-9. URL https://doi.org/10.1038/s41467-025-58532-9.

Hiroki Naganuma, Atish Agarwala, Priya Kasimbeg, and George E. Dahl. What do near-optimal learning rate schedules look like? *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=pEsAMnmq0L. Expert Certification.

Joel Niklaus and Hynek Kydlíček. Smoldata. https://huggingface.co/collections/HuggingFaceFW/smol-data, 2026.

Antonio Orvieto and Robert Gower. In search of adam's secret sauce, 2025. URL https://arxiv.org/abs/2505.21829.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=n6SCkn2QaG.

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained LMOs. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2Oqm2IzTy9.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20. IEEE Press, 2020. ISBN 9781728199986.

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), 2025. URL https://arxiv.org/abs/2505.13416.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017. URL https://arxiv.org/abs/1701.06538.

Jianlin Su. Why is the rms of adam update 0.2? (chinese), Sep 2025. URL https://kexue.fm/archives/11267.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/sutskever13.html.

5 Team, Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu, Rui Lu, Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei, Yean Cheng, Yifan An, Yilin Niu, Yuanhao Wen, Yushi Bai, Zhengxiao Du, Zihan Wang, Zilin Zhu, Bohan Zhang, Bosi Wen, Bowen Wu, Bowen Xu, Can Huang, Casey Zhao, Changpeng Cai, Chao Yu, Chen Li, Chendi Ge, Chenghua Huang, Chenhui Zhang, Chenxi Xu, Chenzheng Zhu, Chuang Li, Congfeng Yin, Daoyan Lin, Dayong Yang, Dazhi Jiang, Ding Ai, Erle Zhu, Fei Wang, Gengzheng Pan, Guo Wang, Hailong Sun, Haitao Li, Haiyang Li, Haiyi Hu, Hanyu Zhang, Hao Peng, Hao Tai, Haoke Zhang, Haoran Wang, Haoyu Yang, He Liu, He Zhao, Hongwei Liu, Hongxi Yan, Huan Liu, Huilong Chen, Ji Li, Jiajing Zhao, Jiamin Ren, Jian Jiao, Jiani Zhao, Jianyang Yan, Jiaqi Wang, Jiayi Gui, Jiayue Zhao, Jie Liu, Jijie Li, Jing Li, Jing Lu, Jingsen Wang, Jingwei Yuan, Jingxuan Li, Jingzhao Du, Jinhua Du, Jinxin Liu, Junkai Zhi, Junli Gao, Ke Wang, Lekang Yang, Liang Xu, Lin Fan, Lindong Wu, Lintao Ding, Lu Wang, Man Zhang, Minghao Li, Minghuan Xu, Mingming Zhao, Mingshu Zhai, Pengfan Du, Qian Dong, Shangde Lei, Shangqing Tu, Shangtong Yang, Shaoyou Lu, Shijie Li, Shuang Li, Shuang-Li, Shuxun Yang, Sibo Yi, Tianshu Yu, Wei Tian, Weihan Wang, Wenbo Yu, Weng Lam Tam, Wenjie Liang, Wentao Liu, Xiao Wang, Xiaohan Jia, Xiaotao Gu, Xiaoying Ling, Xin Wang, Xing Fan, Xingru Pan, Xinyuan Zhang, Xinze Zhang, Xiuqing Fu, Xunkai Zhang, Yabo Xu, Yandong Wu, Yida Lu, Yidong Wang, Yilin Zhou, Yiming Pan, Ying Zhang, Yingli Wang, Yingru Li, Yinpei Su, Yipeng Geng, Yitong Zhu, Yongkun Yang, Yuhang Li, Yuhao Wu, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yuxuan Zhang, Zezhen Liu, Zhen Yang, Zhengda Zhou, Zhongpei Qiao, Zhuoer Feng, Zhuorui Liu, Zichen Zhang, Zihan Wang, Zijun Yao, Zikang Wang, Ziqiang Liu, Ziwei Chai, Zixuan Li, Zuodong Zhao, Wenguang Chen, Jidong Zhai, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-4.5 team: Agentic, reasoning, and coding (arc) foundation models, 2025. URL https://arxiv.org/abs/2508.06471.

Thinking Machines Lab. Inkling: Our open-weights model. https://thinkingmachines.ai/news/introducing-inkling/, 2026. Accessed: 2026-09-23.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *CoRR*, abs/2012.12877, 2020. URL https://arxiv.org/abs/2012.12877.

Kaiyue Wen, Xingyu Dang, Kaifeng Lyu, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them ii: Hyperball optimization, 2026. URL https://arxiv.org/abs/2606.16899.

Yihao Xiao, Jialong Sun, Zitian Gao, Zeming Wei, Chutian Wang, Ran Tao, Jiaye Teng, and Bryan Dai. Hyperball may not be a free lunch, 2026. URL https://arxiv.org/abs/2607.22444.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1Ddp1-Rb.

Zhanpeng Zhou, Wenjie Zhou, Yufei Gu, Juqiu Wang, Guanwei Zhang, Ruobing Xie, Xianbiao Qi, Yushi Bai, Kaifeng Lyu, and Shunyu Yao. From lr to elr: A better heuristic for pretraining dynamics, 2026. URL https://hy.tencent.ai/research/elr.

A MOMENTUM VARIANTS

A.1 NESTEROV MOMENTUM

With Nesterov momentum, the update rules of Scion become

$$\begin{aligned}
\boldsymbol{g}_{t,l} &\leftarrow \nabla_{\theta_l} f_t(\boldsymbol{\theta}_{t-1,l}, \zeta_t) \\
\boldsymbol{m}_{t,l} &\leftarrow (1-\alpha)\boldsymbol{m}_{t-1,l} + \alpha \boldsymbol{g}_{t,l} \\
\boldsymbol{\theta}_{t,l} &\leftarrow (1-\eta)\boldsymbol{\theta}_{t-1,l} + \gamma_l \operatorname{lmo}_l((1-\alpha)\boldsymbol{m}_{t,l} + \alpha \boldsymbol{g}_{t,l}) \\
&= \boldsymbol{\theta}_{t-1,l} + \gamma_l \left(-\lambda_l \boldsymbol{\theta}_{t-1,l} + \operatorname{lmo}_l((1-\alpha)\boldsymbol{m}_{t,l} + \alpha \boldsymbol{g}_{t,l})\right)
\end{aligned}$$

Since the update rule of $\boldsymbol{m}_{t,l}$ remains unchanged, given the same assumptions that the mini-batch gradients become independent with time-independent expected L_2 norm C' at steady state, $\mathbb{E}[\langle \boldsymbol{g}_{t'}, \boldsymbol{g}_t \rangle] = C'^2 \delta_{t't}$, we still have

$$\begin{aligned}
\boldsymbol{m}_t &= (1-\alpha)^k \boldsymbol{m}_{t-k} + \alpha \sum_{i=0}^{k-1} (1-\alpha)^i \boldsymbol{g}_{t-i}\,, \\
\mathbb{E}[\langle \boldsymbol{m}_{t-k}, \boldsymbol{m}_t \rangle] &= \frac{\alpha}{2-\alpha} C'^2 (1-\alpha)^k \text{ for } k \geq 1
\end{aligned}$$

However, update at time t is now $\boldsymbol{u}'_t = -\operatorname{lmo}((1-\alpha)\boldsymbol{m}_t + \alpha \boldsymbol{g}_t)$ instead of $\boldsymbol{u}_t = -\operatorname{lmo}(\boldsymbol{m}_t)$. Again consider the Bias $\operatorname{lmo}_{b_\ell}$ in Table 1 that normalizes the update $\boldsymbol{u}'_t = -\operatorname{lmo}_{b_\ell}((1-\alpha)\boldsymbol{m}_t + \alpha \boldsymbol{g}_t) = \frac{(1-\alpha)\boldsymbol{m}_t + \alpha \boldsymbol{g}_t}{\|(1-\alpha)\boldsymbol{m}_t + \alpha \boldsymbol{g}_t\|_{\mathrm{RMS}}}$. We can derive $\mathbb{E}[||(1-\alpha)\boldsymbol{m}_t + \alpha \boldsymbol{g}_t||_2^2]$ based on independence:

$$\begin{aligned}
\mathbb{E}[||(1-\alpha)\boldsymbol{m}_t + \alpha \boldsymbol{g}_t||_2^2] &= \mathbb{E}[||(1-\alpha)^2 \boldsymbol{m}_{t-1} + \alpha(2-\alpha)\boldsymbol{g}_t||_2^2] \\
&= (1-\alpha)^4 \frac{\alpha}{2-\alpha} C'^2 + \alpha^2 (2-\alpha)^2 C'^2 \\
&= \frac{\alpha}{2-\alpha}(1 + 4\alpha - 6\alpha^2 + 2\alpha^3) C'^2
\end{aligned}$$

So now we expect the TUC of the minibatch to be better characterized by the effective learning rate $\gamma_{\text{eff}} := \gamma \sqrt{\frac{2-\alpha}{\alpha}} (1 + 4\alpha - 6\alpha^2 + 2\alpha^3)^{-\frac{1}{2}}$. Explicitly, if the lmo normalizes the update $||\boldsymbol{u}'_t||_2 = ||\operatorname{lmo}((1-\alpha)\boldsymbol{m}_t + \alpha \boldsymbol{g}_t)|| = C$, denote the normalizing constant $A^2 = \frac{C^2}{\mathbb{E}[||(1-\alpha)\boldsymbol{m}_t + \alpha \boldsymbol{g}_t||_2]^2} = \frac{2-\alpha}{\alpha(1+4\alpha-6\alpha^2+2\alpha^3)} \frac{C^2}{C'^2}$, so

$$\begin{aligned}
\mathbb{E}[\langle \boldsymbol{u}'_{t-k}, \boldsymbol{u}'_t \rangle] &\approx A^2 \mathbb{E}[\langle (1-\alpha)\boldsymbol{m}_{t-k} + \alpha \boldsymbol{g}_{t-k}, (1-\alpha)\boldsymbol{m}_t + \alpha \boldsymbol{g}_t \rangle] \\
&= A^2((1-\alpha)^2 \mathbb{E}[\langle \boldsymbol{m}_{t-k}, \boldsymbol{m}_t \rangle] + \alpha(1-\alpha)\mathbb{E}[\langle \boldsymbol{g}_{t-k}, \boldsymbol{m}_t \rangle]) \\
&= A^2((1-\alpha)^{k+2} \mathbb{E}[\|\boldsymbol{m}_t\|_2^2] + \alpha^2 (1-\alpha)^{k+1} \mathbb{E}[\|\boldsymbol{g}_t\|_2^2]) \\
&= A^2 (1-\alpha)^k ((1-\alpha)^2 \mathbb{E}[\|\boldsymbol{m}_t\|_2^2] + \alpha^2 (1-\alpha)\mathbb{E}[\|\boldsymbol{g}_t\|_2^2]) \\
&= A^2 (1-\alpha)^k ((1-\alpha)^2 \frac{\alpha}{2-\alpha} C'^2 + \alpha^2 (1-\alpha) C'^2) \\
&= C^2 (1-\alpha)^k \frac{2-\alpha}{\alpha(1+4\alpha-6\alpha^2+2\alpha^3)} ((1-\alpha)^2 \frac{\alpha}{2-\alpha} + \alpha^2(1-\alpha)) \\
&= C^2 (1-\alpha)^k \frac{(1-\alpha)^2 + \alpha(2-\alpha)(1-\alpha)}{1+4\alpha-6\alpha^2+2\alpha^3} = C^2 (1-\alpha)^k \frac{1-2\alpha+\alpha^2+\alpha(2-3\alpha+\alpha^2)}{1+4\alpha-6\alpha^2+2\alpha^3} \\
&= C^2 (1-\alpha)^k \frac{1-2\alpha^2+\alpha^3}{1+4\alpha-6\alpha^2+2\alpha^3} = \kappa C^2 (1-\alpha)^k
\end{aligned}$$

where $\kappa = \frac{1-2\alpha^2+\alpha^3}{1+4\alpha-6\alpha^2+2\alpha^3}$. The model parameter's update rule in terms of $\boldsymbol{u}'_t$ with Nesterov momentum is the same as the update rule in terms of $\boldsymbol{u}_t$ without, so now we have

$$\begin{aligned}
\boldsymbol{\theta}_t &= (1-\eta)\boldsymbol{\theta}_{t-1} - \gamma \boldsymbol{u}'_t \\
&= -\gamma \sum_{i=0}^{\infty} (1-\eta)^i \boldsymbol{u}'_{t-i} \\
\boldsymbol{\theta}_{t-1} &= -\gamma \sum_{i=0}^{\infty} (1-\eta)^i \boldsymbol{u}'_{t-1-i} \\
\mathbb{E}[\langle \boldsymbol{\theta}_{t-1}, \boldsymbol{u}'_t \rangle] &= -\gamma \sum_{i=0}^{\infty} (1-\eta)^i \mathbb{E}[\langle \boldsymbol{u}'_{t-1-i}, \boldsymbol{u}'_t \rangle] \\
&= -\kappa\gamma C^2 \sum_{i=0}^{\infty} (1-\eta)^i (1-\alpha)^{i+1} \\
&= -\kappa\gamma C^2 (1-\alpha) \sum_{i=0}^{\infty} (1-\eta)^i (1-\alpha)^i \\
&= -\frac{\kappa\gamma C^2 (1-\alpha)}{1-(1-\eta)(1-\alpha)} = -\frac{\kappa\gamma C^2 (1-\alpha)}{\eta+\alpha-\alpha\eta}
\end{aligned}$$

Since independent weight decay coefficient $\eta = \gamma\lambda$:

$$\mathbb{E}[||\boldsymbol{\theta}_t||^2] = \mathbb{E}[(1-\eta)^2 ||\boldsymbol{\theta}_{t-1}||^2 + \gamma^2 ||\boldsymbol{u}'_t||^2 - 2\gamma(1-\eta)\langle \boldsymbol{\theta}_{t-1}, \boldsymbol{u}'_t \rangle]$$

With $||\boldsymbol{u}'_t||^2 = C^2$ and the expression above, at steady state $\mathbb{E}[||\boldsymbol{\theta}_t||^2] = \mathbb{E}[||\boldsymbol{\theta}_{t-1}||^2]$:

$$(2\eta - \eta^2)\mathbb{E}[||\boldsymbol{\theta}_t||^2] = \gamma^2 C^2 (1 + 2\kappa \frac{(1-\eta)(1-\alpha)}{\eta+\alpha-\alpha\eta})$$

$$\begin{aligned}
\mathbb{E}[||\boldsymbol{\theta}_t||^2] &= \frac{\gamma^2 C^2}{2\eta-\eta^2} \left(\frac{\alpha(1+4\alpha-6\alpha^2+2\alpha^3) + 2(1-\alpha)(1-2\alpha^2+\alpha^3) + O(\eta)}{\alpha(1+4\alpha-6\alpha^2+2\alpha^3) + O(\eta)} \right) \\
&= \frac{\gamma^2 C^2}{2\eta-\eta^2} \left(\frac{\alpha + 4\alpha^2 - 6\alpha^3 + 2\alpha^4 + 2 - 4\alpha^2 + 2\alpha^3 - 2\alpha + 4\alpha^3 - 2\alpha^4) + O(\eta)}{\alpha(1+4\alpha-6\alpha^2+2\alpha^3) + O(\eta)} \right) \\
&= \frac{\gamma^2 C^2}{2\eta-\eta^2} \left(\frac{2-\alpha+O(\eta)}{\alpha(1+4\alpha-6\alpha^2+2\alpha^3) + O(\eta)} \right)
\end{aligned}$$

Typically $\eta \ll \alpha \leq 1$. Ignore $O(\eta^2)$ and $O(\eta^3)$ terms of the denominator and $O(\eta)$ terms of the numerator, we get

$$\mathbb{E}[||\boldsymbol{\theta}_t||^2] \approx \frac{\gamma^2 C^2}{2\eta} \left(\frac{2-\alpha}{\alpha(1+4\alpha-6\alpha^2+2\alpha^3)} \right) = \frac{\gamma_{\text{eff}}^2 C^2}{2\eta} \tag{7}$$

$$= \frac{\gamma C^2}{2\lambda} \left(\frac{2-\alpha}{\alpha(1+4\alpha-6\alpha^2+2\alpha^3)} \right) \tag{8}$$

where now $\gamma_{\text{eff}} :- \gamma\sqrt{\frac{2-\alpha}{\alpha}}(1+4\alpha-6\alpha^2+2\alpha^3)^{-\frac{1}{2}}$ as expected.

A.2 Trace momentum

In some variants of Muon (Jordan et al., 2024; Kimi Team et al., 2026) the momentum is computed as $\boldsymbol{m}'_{t,l} \leftarrow \mu\boldsymbol{m}'_{t-1,l} + \boldsymbol{g}_{t,l}$ (sometimes referred to as the "trace", DeepMind et al. (2020)), so the update rule becomes of the form

$$\begin{aligned}
\boldsymbol{g}_{t,l} &\leftarrow \nabla_{\theta_l} f_t(\boldsymbol{\theta}_{t-1,l}, \zeta_t) \\
\boldsymbol{m}'_{t,l} &\leftarrow \mu\boldsymbol{m}'_{t-1,l} + \boldsymbol{g}_{t,l} \\
\boldsymbol{\theta}_{t,l} &\leftarrow (1-\eta)\boldsymbol{\theta}_{t-1,l} + \gamma_l \,\text{lmo}_l(\boldsymbol{m}'_{t,l}) \\
&= \boldsymbol{\theta}_{t-1,l} + \gamma_l \left(-\lambda_l \boldsymbol{\theta}_{t-1,l} + \text{lmo}_l(\boldsymbol{m}'_{t,l})\right)
\end{aligned}$$

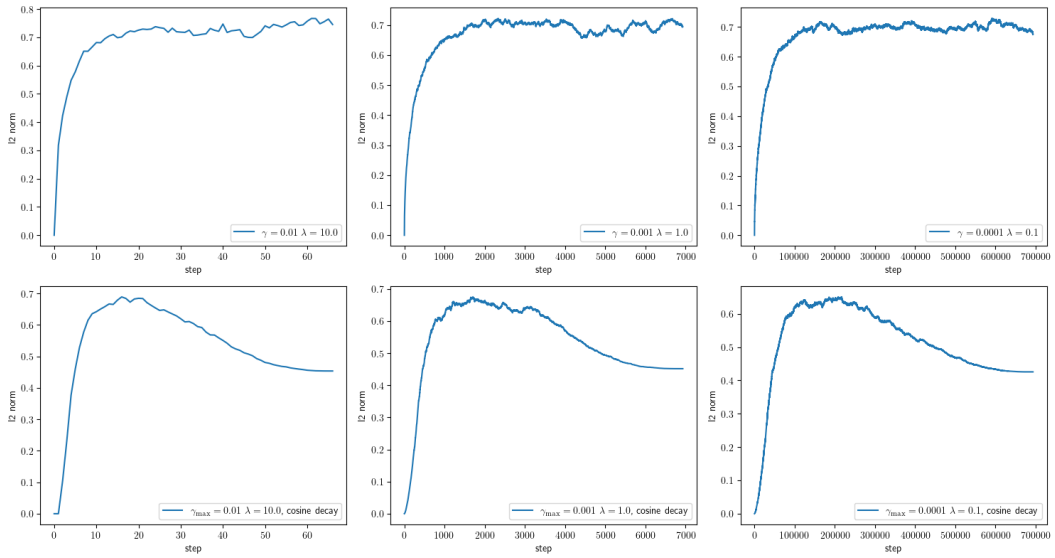


Figure 7: Numerical simulations of the system described by Eq. 9 where θ is a vector of length 10^3. $\mathbb{E}[\theta_t^2] = \frac{\gamma C}{2\lambda} = \frac{1}{2000}$ for each element, so the expected L_2 norm of the vector is ≈ 0.71 if we keep the learning rate constant (upper) as expected. If we apply cosine learning rate decay (lower), weight norm decreases towards the end. Here we consistently simulate the system for 10 half-lives $t_{1/2} = -\frac{\log 2}{\log(1-\gamma\lambda)}$, with $0.5\,t_{1/2}$ of linear-warmup and $9.5\,t_{1/2}$ of cosine learning rate decay, so the behavior of the systems looks identical despite 4 orders of magnitudes of difference in scale.

Assume again that $\mathbb{E}[\langle \boldsymbol{g}_{t'}, \boldsymbol{g}_t \rangle] = C'^2 \delta_{t't}$ at steady state. We can see that the update rule is equivalent to the case with exponential moving average (EMA) momentum, $\alpha = 1 - \mu$, and $\mathbb{E}[\langle \boldsymbol{g}_{t'}, \boldsymbol{g}_t \rangle] = \frac{C'^2}{\alpha^2} \delta_{t't}$. Neither the effective learning rate nor the steady-state weight norm depends on the norm of the minibatch gradient at steady state, however, so they are both identical to their EMA momentum counterpart with $\gamma_{\text{eff}} := \gamma \sqrt{\frac{1+\mu}{1-\mu}}$ and $\mathbb{E}[||\boldsymbol{\theta}_t||^2] \approx \frac{\gamma_{\text{eff}}^2 C^2}{2\eta}$. Similarly, in the case of trace and Nesterov momentum with update rule

$$\begin{aligned}
\boldsymbol{g}_{t,l} &\leftarrow \nabla_{\theta_l} f_t(\boldsymbol{\theta}_{t-1,l}, \zeta_t) \\
\boldsymbol{m}'_{t,l} &\leftarrow \mu \boldsymbol{m}'_{t-1,l} + \boldsymbol{g}_{t,l} \\
\boldsymbol{\theta}_{t,l} &\leftarrow (1-\eta)\boldsymbol{\theta}_{t-1,l} + \gamma_l \operatorname{lmo}_l(\mu \boldsymbol{m}'_{t,l} + \boldsymbol{g}_{t,l}) \\
&= \boldsymbol{\theta}_{t-1,l} + \gamma_l \left(-\lambda_l \boldsymbol{\theta}_{t-1,l} + \operatorname{lmo}_l(\mu \boldsymbol{m}'_{t,l} + \boldsymbol{g}_{t,l})\right)
\end{aligned}$$

The effective learning rate is $\gamma_{\text{eff}} := \gamma \sqrt{\frac{1+\mu}{1-\mu}} (1 + 2\mu - 2\mu^3)^{-\frac{1}{2}}$ and $\mathbb{E}[||\boldsymbol{\theta}_t||^2]$ remains $\frac{\gamma_{\text{eff}}^2 C^2}{2\eta}$.

B Numerical Simulations

B.1 Weight norm evolution with and without learning rate schedule

Consider the following system where $\boldsymbol{\theta}$ is initialized as $\boldsymbol{\theta}_0 = 0$:

$$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \gamma \left(\lambda \boldsymbol{\theta}_{t-1} + \mathcal{N}(0, 1)\right) \tag{9}$$

It turns out that this simple system is sufficient to replicate the weight norm behavior towards the end of the cosine learning rate decay, suggesting that the nature of the optimizer is not fundamental to such phenomena (Fig. 7).

B.2 Steady-state weight norm with normalized update

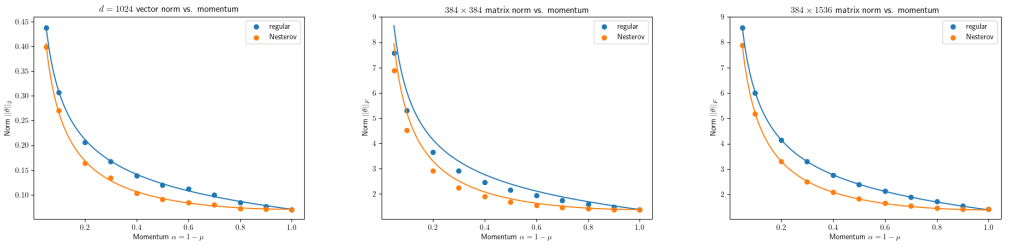
We run numerical simulations with Scion update rules and unit Gaussian random vectors / matrices as mock minibatch gradients:

$$\begin{aligned}\boldsymbol{g}_t &\leftarrow \mathcal{N}(0,\,1)\\ \boldsymbol{m}_t &\leftarrow (1-\alpha)\boldsymbol{m}_{t-1} + \alpha \boldsymbol{g}_t\\ \boldsymbol{\theta}_t &\leftarrow \boldsymbol{\theta}_{t-1} + \gamma\left(-\lambda\boldsymbol{\theta}_{t-1} + \mathrm{lmo}(\boldsymbol{m}_t)\right)\end{aligned}$$

where $\boldsymbol{\theta}_0 = 0, \boldsymbol{m}_0 = \boldsymbol{g}_0$ and its Nesterov momentum counterpart:

$$\begin{aligned}\boldsymbol{g}_t &\leftarrow \mathcal{N}(0,\,1)\\ \boldsymbol{m}_t &\leftarrow (1-\alpha)\boldsymbol{m}_{t-1} + \alpha \boldsymbol{g}_t\\ \boldsymbol{\theta}_t &\leftarrow \boldsymbol{\theta}_{t-1} + \gamma\left(-\lambda\boldsymbol{\theta}_{t-1} + \mathrm{lmo}((1-\alpha)\boldsymbol{m}_t + \alpha\boldsymbol{g}_t)\right)\end{aligned}$$

For simplicity, we use $\mathrm{lmo}(\boldsymbol{m}_t) = -\frac{\boldsymbol{m}_t}{||\boldsymbol{m}_t||_2}$ for vector and $\mathrm{lmo}(\boldsymbol{m}_t) = -\boldsymbol{U}\boldsymbol{V}^\top$ (reduced SVD) for matrix since the lmo's in practice only differ by constant factors (RowNorm / ColNorm / Bias and Spectral, respectively). We use $\gamma = 0.001, \lambda = 0.1$ and compare the final weight norm $||\boldsymbol{\theta}||_2$ or $||\boldsymbol{\theta}||_F$ to the prediction $\frac{\gamma_{\text{eff}}^2 C^2}{2\eta}, \eta = \gamma\lambda$, effective learning rate $\gamma_{\text{eff}} := \gamma\sqrt{\frac{2-\alpha}{\alpha}}$ (standard momentum) or $\gamma_{\text{eff}} := \gamma\sqrt{\frac{2-\alpha}{\alpha}}(1 + 4\alpha - 6\alpha^2 + 2\alpha^3)^{-\frac{1}{2}}$ (Nesterov momentum) after 10 half-lives $t_{1/2} = -\frac{\log 2}{\log(1-\eta)}$ (Fig. 8). Numerical simulations and predictions are in excellent agreement for vectors and $d_{\text{in}} = 4d_{\text{out}}$ matrices (standard in transformer MLP layers) but deviate up to $\approx 10\%$ for square matrices. We do not have good explanations for the latter result and therefore still look for better approximations.



$d = 1024$ vector $\quad$ $d_{\text{out}} = d_{\text{in}} = 384$ matrix $\quad$ $d_{\text{out}} = 384, d_{\text{in}} = 1536$ matrix

Figure 8: Final weight norm after 10 half-lives of numerical simulation vs. steady-state prediction for $d = 1024$ vector (left), $d_{\text{out}} = d_{\text{in}} = 384$ matrix (middle), and $d_{\text{out}} = 384, d_{\text{in}} = 1536$ matrix (right).

C Modded-NanoGPT with RowNorm

We test optimal γ_{eff} transfer on a Modded-NanoGPT variant with RowNorm and without weight tying. Without weight tying, the embedding is considered an input layer and optimized with RowNorm and corrected weight decay parameterized with C^2 along with the hidden layers. The output ("unembedding") layer is still optimized with the Sign lmo. Fig. 9 shows again that the optimal effective learning rate transfers within a factor of $\sqrt{2}$ while the optimal learning rate varies by almost a factor of 10 as we vary the momentum 50-fold and stable input + hidden layer weight norm as expected.

D Nested Tuning of Momentum Schedule

We test both exponential-decaying momentum $\alpha \propto e^{-ct}$ (Defazio, 2026) and log-time momentum $\alpha \propto \frac{\delta}{\delta+t}$ (Ferbach et al., 2026) schedules for training the 124M Modded-NanoGPT on the full FineWeb-Edu-100BT-shuffle. The nested tuning finds the optimal end momentum $\alpha_T = \alpha_0 2^{-\frac{5}{3}n'}, n' \in \mathbb{N}$ for each start momentum $\alpha_0 = \{1, 2, 5\} \times 10^{-n}, n \in \mathbb{N}$ until it finds a local optimum $(\alpha_{0,o}, \alpha_{T,o})$ that also outperforms (α_0, α_T) for both the previous and the next value of α_0 for all values of α_T we consider. The exponential-decaying momentum result suggests that the

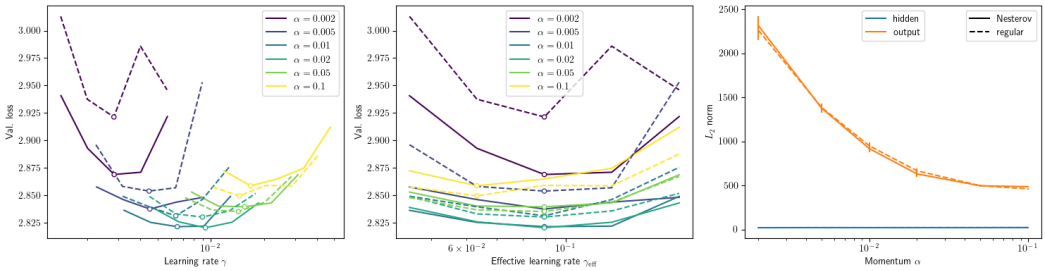


Figure 9: Final val. loss of the RowNorm Modded-NanoGPT after training on $3/10$ of FineWeb-Edu-100BT-shuffle with various Nesterov (solid) and regular (dashed) momentum α vs. the learning rate (left) and effective learning rate (middle) and the final weight norm of the input + hidden layers and output layer (right).

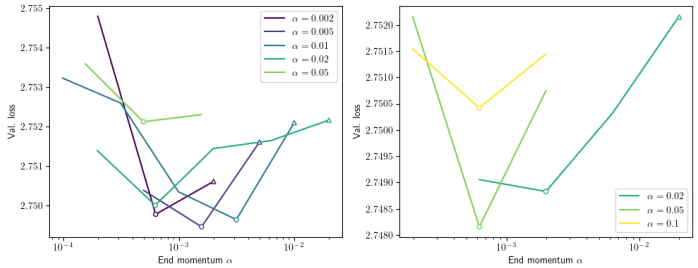


Figure 10: Final val. loss of the 124M Modded-NanoGPT after training on the full FineWeb-Edu-100BT-shuffle vs. the end momentum, color-coded by the start momentum, exponential decay (left) and log-time (right). Triangle markers highlight the data points with constant momentum and circle markers highlight the minimum for the respective start momentum.

optimal constant momentum $\alpha \leq 0.002$ but log-time momentum reaches the best final val. loss with $(\alpha_0, \alpha_T) = (0.05, 0.00062)$ (Fig. 10).

E BETAS' EFFECT ON THE WEIGHT DECAY AND STEADY-STATE NORM FOR ADAMC

We train a ViT-S/16 on the ImageNet-1k dataset (Russakovsky et al., 2015) for 90 epochs with AdamC and $\beta_1 = \beta_2 = 0.99$ instead of $(\beta_1, \beta_2) = (0.9, 0.999)$ of the main experiment, partially motivated by Orvieto & Gower (2025) (Fig. 11). As predicted, changing betas has no effect on the weight decay and steady-state norm.

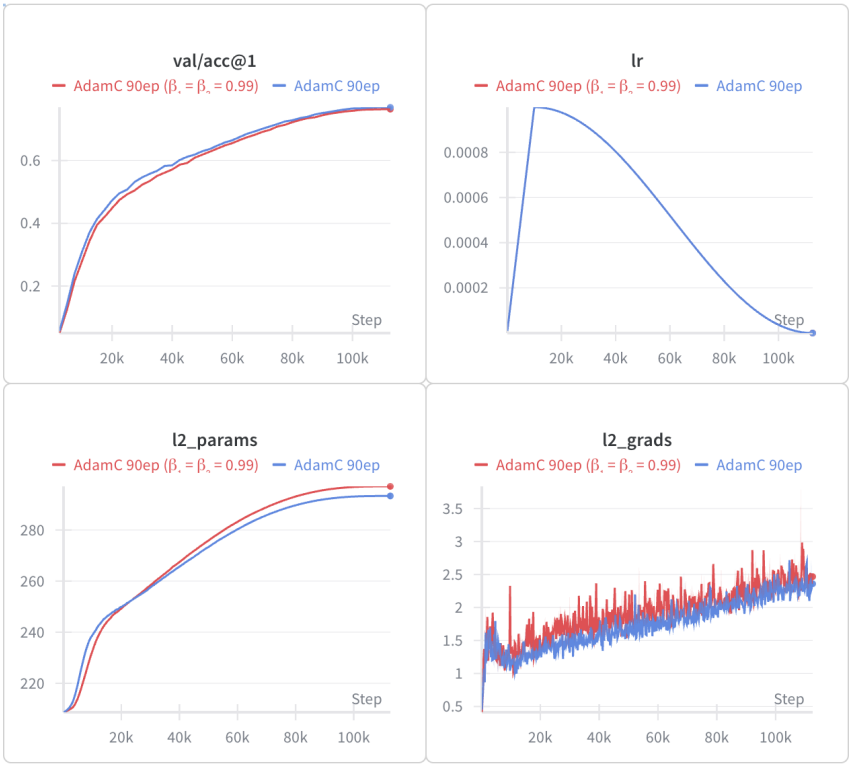


Figure 11: Training a ViT-S/16 on ImageNet-1k for 90 epochs, AdamC with $\beta_1 = \beta_2 = 0.99$ vs. AdamC with $(\beta_1, \beta_2) = (0.9, 0.999)$. Changing the beta values has almost no effects on the weight norm.