# HYPERBALL OPTIMIZATION FOR LLM PRETRAINING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Matrix-based optimizers such as Muon can substantially speed up language model pretraining, but their gains over AdamW are observed to shrink as model size and data scale grow when using standard constant decoupled weight decay. We propose **Hyperball**, a simple optimizer wrapper that addresses this issue. The method is motivated by prior theory showing that weight decay drives the weight norm toward an equilibrium, eventually making the weights move only on a hypersphere of fixed radius. With Hyperball, we explicitly control this behavior: given a base optimizer such as Adam or Muon, Hyperball sets the Frobenius norms of weight matrices and their corresponding optimizer updates to fixed constants. On Qwen3 style models up to 1.2B parameters, Muon Hyperball achieves 20–30% token equivalent speedup over weight decay baselines. Hyperball also improves learning rate transfer across widths and depths compared to decoupled weight decay.

## 1 INTRODUCTION

Previous work (Wen et al., 2025) observed that the speedups of matrix-based optimizers such as Muon (Jordan et al., 2024; Liu et al., 2025) over AdamW (Loshchilov & Hutter, 2019) shrink from roughly 30% to about 10% as model size and data scale grow. This motivates a simple question: *can we keep these optimizer speedups at higher compute?*

In this paper, we propose **Hyperball** as a simple solution to the above question. Given a base optimizer (e.g., Adam, Muon), Hyperball acts as an optimizer wrapper that transforms it into a new optimizer that maintains a constant weight norm and a constant update norm. We refer to this new optimizer as the Hyperball variant of the base optimizer, such as Adam Hyperball (AdamH), Muon Hyperball (MuonH).

To explain our motivation, we first review how weight decay affects training dynamics in modern architectures. Let $W_t$ be a weight matrix at step $t$, $u_t$ be the update provided by a base optimizer before learning rate multiplication, $\eta_t$ be the learning rate, and $\lambda$ be the weight decay coefficient. The standard update rule with decoupled weight decay (Loshchilov & Hutter, 2019) is $W_{t+1} = (1 - \eta_t\lambda)\, W_t - \eta_t\, u_t$. Here $-\eta_t u_t$ typically increases the weight norm, while $(1 - \eta_t\lambda)W_t$ shrinks it toward zero at each step. Previous work (Li et al., 2020; Roburin et al., 2020; Wan et al., 2021; Kosson et al., 2024a) considered modern architectures with normalization layers (Vaswani et al., 2017; Xiong et al., 2020), where the loss function is approximately scale-invariant with respect to many of its parameters $W$: rescaling these weights by a factor $c > 0$ leaves the loss approximately unchanged, i.e., $L(cW) \approx L(W)$. In this setting, they revealed that the two competing effects of $-\eta_t u_t$ and $(1 - \eta_t\lambda)W_t$ on weight norms can reach an equilibrium, eventually making the weights move only on a hypersphere of fixed radius. Meanwhile, the update size of the weight direction on this hypersphere, referred to as the *angular step size*, also eventually stabilizes to a constant.

However, reaching this equilibrium can take a long time. The continuous change in weight norm also complicates the training dynamics and potentially affects the scalability of optimizers. If so, *why not just remove weight decay, fix the weight norm, and directly control the angular step size?*

Motivated by this idea, Hyperball replaces the soft control of weight decay on weight norm with an explicit constraint and normalizes the optimizer update. Let $\|X\|_{\mathrm{F}} = (\sum_{ij} X_{ij}^2)^{1/2}$ be the Frobenius norm of a matrix $X$, let $R = \|W_0\|_{\mathrm{F}}$ be the initial Frobenius norm of the parameter matrix, and let $\mathrm{Normalize}(X) := X/\|X\|_{\mathrm{F}}$ be Frobenius normalization. The Hyperball update is $W_{t+1} = R \cdot \mathrm{Normalize}(W_t - \eta_t R \cdot \mathrm{Normalize}(u_t))$. Geometrically, Hyperball constrains the optimization trajectory to a hypersphere with radius $R$ (fig. 1): each step moves a distance $\eta_t R$



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

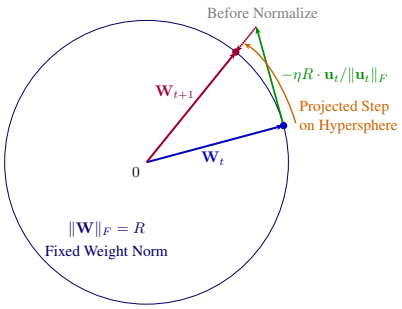


Figure 1: In Hyperball, each step moves a distance $\eta_t R$ along $-\mathrm{Normalize}(u_t)$ and is then projected back to the hypersphere of radius $R$, keeping the weight norm and angular step size fixed.

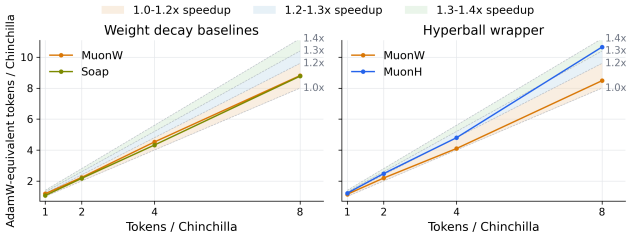


Figure 2: Token equivalent speedup over the AdamW scaling law at 1.2B parameters across Chinchilla ratios $1\times$–$8\times$, without QK-Norm (left, adapted from Wen et al. (2025)) and with QK-Norm (right). MuonW alone gains $\approx 10\%$ at this scale, while MuonH sustains 20–30% speedup that grows with training duration.

along the normalized update direction $-\mathrm{Normalize}(u_t)$ and is immediately projected back onto the hypersphere. In this way, the weight norm stays fixed, and $\eta_t$ directly sets the effective learning rate, defined as the unprojected update size relative to the weight norm. The actual angular step size also depends on the update direction.

Here the base update $u_t$ can come from any optimizer. In this paper, we focus on Adam Hyperball (AdamH) and Muon Hyperball (MuonH). We apply Hyperball to Transformer weight matrices and use Adam for embeddings, normalization gains, and other parameters whose norm carries semantic information. On 1.2B parameter Qwen3 style models (Yang et al., 2025), MuonH achieves 20–30% token-equivalent speedup over AdamW, whereas MuonW gives only about 10% at this scale. Across depth and width sweeps, Hyperball also maintains the best learning rate window better than the baseline: the maximal drift is about $1.4\times$ for AdamH and MuonH, compared with 2–$4\times$ for AdamW and MuonW baselines.

The rest of the paper is organized as follows. We describe Hyperball in detail in Section 2, and provide our main experiments in Section 3. We then provide a dynamical analysis of the weight norm and angular step size that motivate this design in Section 4, along with explanations of several related phenomena. We discuss related work in Section 5 and Section D.

## 2 Method

**Definition.** Let $W_t$ be the parameter matrix at step $t$, $u_t$ be the base optimizer update for this matrix (for example, Adam's preconditioned update (Kingma & Ba, 2015) or Muon's matrix sign momentum update (Jordan et al., 2024; Liu et al., 2025)), $\eta_t$ be the normal Hyperball learning rate, $R > 0$ be the fixed radius, and $\mathrm{Normalize}(X) := X/\|X\|_{\mathrm{F}}$ be Frobenius normalization. For each constrained matrix $W_0 \in \mathbb{R}^{d_{\mathrm{out}} \times d_{\mathrm{in}}}$, we initialize entries with standard deviation $1/\sqrt{d_{\mathrm{in}}}$ and set the radius once as $R = \|W_0\|_{\mathrm{F}}$. The Hyperball update is

$$W_{t+1} \;=\; R \cdot \mathrm{Normalize}\Big(\Big(W_t - \eta_t\, R \cdot \mathrm{Normalize}(u_t)\Big)\Big)\,. \tag{1}$$

Thus Hyperball takes an unconstrained optimizer step with norm $\eta_t R$, followed by radial renormalization to radius $R$ (algorithm 1, fig. 1; the exact displacement is given in section A.1). Writing $\widehat{W}_t := W_t/\|W_t\|_{\mathrm{F}}$, we define the angular step size as $\eta_t^{\mathrm{ang}} := \left\|\widehat{W}_{t+1} - \widehat{W}_t\right\|_{\mathrm{F}}$, the chord distance on the unit Frobenius sphere.

**Where to apply the constraint.** We apply Hyperball to attention and MLP weight matrices in a prenorm Transformer (Vaswani et al., 2017; Xiong et al., 2020). Embeddings, normalization gains, and other scalar parameters are updated with a standard optimizer (Adam in our experiments), since for these parameters the norm can carry more information.

**Algorithm 1** Hyperball wrapper for a parameter matrix $W$

**Input:** parameter matrix $W_t$, base optimizer $\mathcal{O}$, optimizer state $\mathcal{S}_t$, radius $R$, schedule $\{\eta_t\}$
Compute base optimizer update $u_t, S_{t+1} \leftarrow \mathcal{O}(\nabla_{W_t} L(W_t), \mathcal{S}_t)$
Set normalized update direction $\widehat{u}_t \leftarrow \mathrm{Normalize}(u_t)$
Take unprojected step $\widetilde{W}_{t+1} \leftarrow W_t - \eta_t \, R \, \widehat{u}_t$
Project to radius $R$: $W_{t+1} \leftarrow R \cdot \mathrm{Normalize}\left(\widetilde{W}_{t+1}\right)$

**Discussion of the design.** A natural alternative is a spectral norm constraint, in the spirit of the spectral condition of Yang et al. (2023) and of SSO (Xie et al., 2026), which would fix the operator norm $\|W_t\|_{\mathrm{op}}$ and normalize the update in operator norm. We use the Frobenius version in (1) because projection onto the Frobenius sphere costs $O(N^2)$ per matrix whereas exact spectral projection requires an SVD and costs $O(N^3)$, and because the mechanism in section 4 is stated in Frobenius norm; when the stable rank $\|W\|_{\mathrm{F}}^2 / \|W\|_{\mathrm{op}}^2$ is large, as observed empirically for LLM weight matrices (Liu et al., 2025, Appendix F), the two constraints behave similarly (section A.2).

## 3 Hyperball Experiments

### 3.1 Setup

Unless noted otherwise, we use a Qwen3 style decoder only architecture (Yang et al., 2025) with QK-Norm (Henry et al., 2020), trained on a mixture of DCLM-baseline (Li et al., 2024), StarCoder (Li et al., 2023), and ProofPile 2 (Azerbayev et al., 2023) and FineWeb-Edu (Penedo et al., 2024). We compare Adam (Kingma & Ba, 2015) and Muon (Jordan et al., 2024; Liu et al., 2025) with decoupled weight decay (AdamW and MuonW) against their Hyperball variants AdamH and MuonH. For the speedup metric, we fit a scaling law to AdamW across Chinchilla ratios (Hoffmann et al., 2022) $\{1\times, 2\times, 4\times, 8\times\}$ and report, for each method's final loss, the token ratio $\tau = N_{\mathrm{AdamW}}/N_{\mathrm{method}}$ that AdamW would need to match it. For learning rate transfer, we sweep a multiplicative grid (ratio $\sqrt{2}$) at each scale $s$, define $\eta^\star(s) := \arg\min_{\eta_k} \ \mathrm{ValLoss}(s, \eta_k; T)$, and report $\mathrm{Drift} := \max_s \eta^\star(s) \,/\, \min_s \eta^\star(s)$.

### 3.2 End-to-end speedup

In the weight decay baseline setting adopted from Wen et al. (2025), 1.2B parameter Qwen3 style models are trained over Chinchilla ratios $1\times$–$8\times$, and MuonW alone yields $\approx 10\%$ token equivalent speedup over the AdamW scaling law. MuonH instead sustains 20–30% speedup, and the gap *grows* with training duration (fig. 2). Qualitatively, Hyperball starts slightly worse but overtakes WD as the learning rate decays. On the Marin speedrun benchmark (FineWeb-Edu, $1\times$ Chinchilla), AdamH and MuonH match WD baselines that are $\approx 10\%$ larger (fig. 3, left). In Marin Ferries, scaling MuonH to 8B parameters yields a further 0.04 loss improvement over the AdamW baseline, with both runs using manually chosen hyperparameters (fig. 3, middle and right).

**Effective learning rate schedule is the reason behind the speedup.** We define the *effective learning rate* as the update size relative to the weight norm, which serves as a proxy measure of the angular update size on the hypersphere. For optimizers with WD, the effective learning rate is $\eta_t \, \|u_t\|_{\mathrm{F}} \,/\, \|W_t\|_{\mathrm{F}}$. Figure 10 in section C compares this proxy for a WD run with Hyperball's prescribed linear decay, after normalizing each curve by its peak value. With WD, shrinking weight norms partly offset learning rate decay, delaying the reduction in relative update size. Hyperball's relative update size follows the prescribed linear decay directly. Follow-up work (Xiao et al., 2026; Zhou et al., 2026; Liu et al., 2026; Yuan et al., 2026) has found that this change in effective learning rate schedule likely contributes to Hyperball's late-training speedup, as matching effective learning rates can largely align the loss trajectories of the Hyperball and weight decay variants.

On the public modded-nanogpt Track 3 optimization benchmark, which fixes the model and data and measures optimizer progress by step count, the corresponding WD baselines reach average validation loss 3.2790 in 5625 steps for the single run AdamW baseline, 3.2790 in 3325 steps for MuonW (tuned), and 3.2789 in 3250 steps for NorMuonW (Li et al., 2025). AdamH reaches average

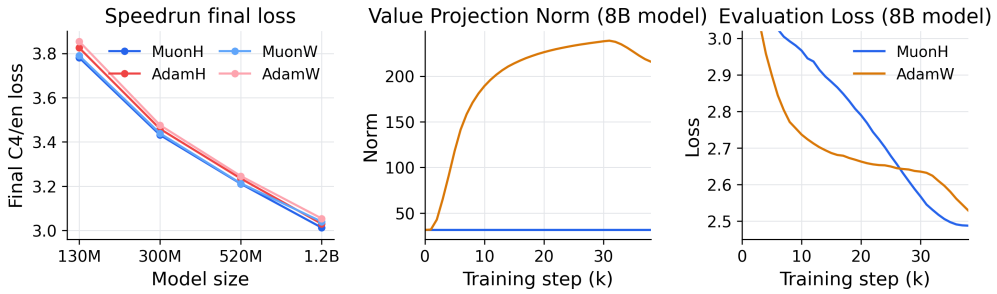


Figure 3: Additional Marin benchmarks. Left: final C4/en loss (Raffel et al., 2020) on the FineWeb-Edu speedrun benchmark at $1\times$ Chinchilla. Middle and right: 8B model comparison over 159B tokens. MuonH fixes the layer 9 value projection matrix norm and finishes 0.04 lower than the AdamW baseline.

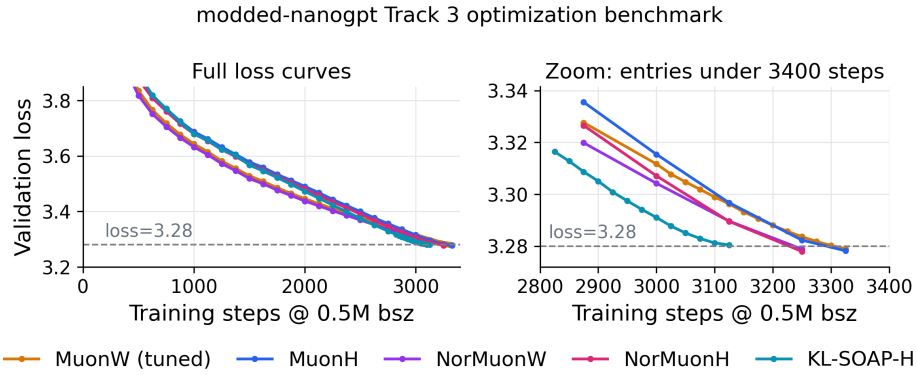


Figure 4: The modded-nanogpt Track 3 optimization benchmark: average validation loss from the public Track 3 logs versus training step (lower and further left is better). Left: full trajectories. Right: zoom on the 3.27–3.28 band for entries reaching it within 3400 steps. KL-SOAP-H denotes KL-SOAP (Lin et al., 2026) combined with Hyperball.

validation loss 3.2741 in 4875 steps. MuonH reaches average validation loss 3.2782 in 3325 steps, NorMuonH reaches average validation loss 3.2778 in 3250 steps, and KL-SOAP-H (KL-SOAP with Hyperball) (Lin et al., 2026) reaches average validation loss 3.2780 in 3125 steps (fig. 4) (Jordan, 2026). This result shows that Hyperball is not tied to Muon or Adam: it reaches the target loss in the fewest steps in this comparison when applied to KL-SOAP.

### 3.3 Hyperparameter transfer

By construction, Hyperball fixes $\|W_t\|_{\mathrm{F}} = R$ and uses unit Frobenius update directions, so $\eta_t$ directly sets the effective learning rate. The optimal learning rate should therefore be near constant following Yang et al. (2023). We test this in two sweeps with 10B tokens per run. These transfer runs use a hybrid normalization architecture variant (Zhuo et al., 2025), with QK-Norm enabled.

For depth scaling at fixed hidden dimension $d = 128$ and $L \in \{4, \ldots, 512\}$, the maximal drift of the optimal learning rate is $\approx 1.4\times$ for AdamH and MuonH, versus $\approx 3\times$ for AdamW and $\approx 4\times$ for MuonW even at $L = 512$ (fig. 5, top). For width scaling at fixed depth $L = 4$ and $d \in \{128, \ldots, 2048\}$, the same $\approx 1.4\times$ drift holds for both Hyperball variants (fig. 5, bottom).

### 3.4 Overtrained Setting

We also test Hyperball in an overtrained data scaling setting. For a 130M parameter model, we train MuonW and MuonH over token budgets from 1B to 128B and sweep the learning rate at each budget. Both MuonW and MuonH use the same hybrid normalization architecture variant as in the previous section. MuonH attains lower best C4 validation loss across the full range (fig. 11 in section C). Fitting $L(N) = L_\infty + AN^{-\alpha}$ to the best loss curve gives $L_\infty = 3.065$ for MuonH versus 3.079 for MuonW.

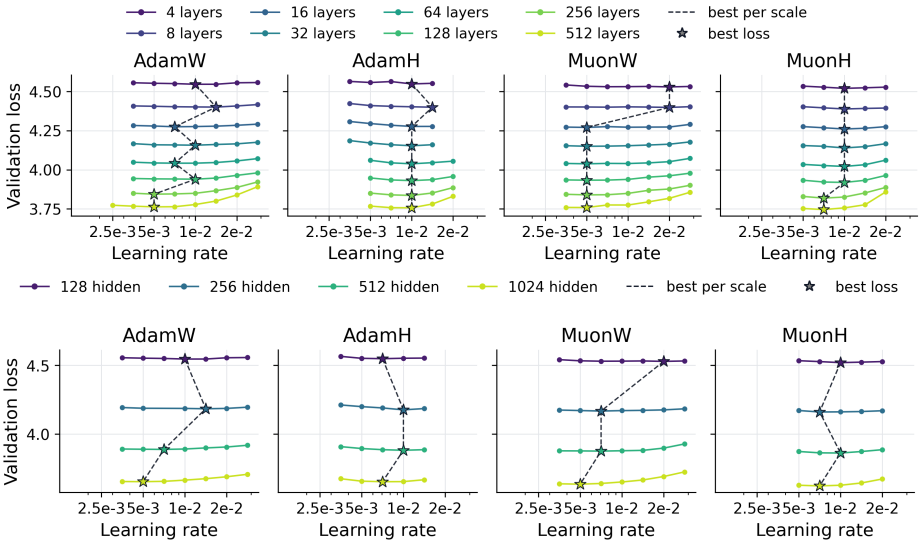


Figure 5: Learning rate transfer across depth (top: fixed $d = 128$, $L \in [4, 512]$) and width (bottom: fixed $L = 4$, $d \in [128, 2048]$), 10B tokens per run; stars mark the best learning rate. Hyperball variants reduce the optimal learning rate drift to $\approx 1.4\times$, versus 2–4$\times$ for AdamW and MuonW.

# 4 Theory

## 4.1 Expressivity

In normalized networks, fixing the Frobenius norm of a weight matrix does not limit representation power, because a trainable normalization gain can absorb the scale of the weight matrix. Let $h$ be the hidden state, $W$ be a weight matrix, and $\gamma$ be the RMSNorm gain (Zhang & Sennrich, 2019). For a linear map placed after a layer norm, $f(h; W, \gamma) = W\,(\gamma \odot \mathrm{RMSNorm}(h))$, and for any scalar $c > 0$ the joint rescaling $(W, \gamma) \mapsto (cW, \gamma/c)$ leaves the represented function unchanged, $f(h; cW, \gamma/c) = f(h; W, \gamma)$. Thus constraining $\|W\|_{\mathrm{F}}$ need not reduce the represented function class. Hyperball changes the optimization geometry without significantly limiting function space.

## 4.2 Optimization

Let $W$ be a parameter block and $\mathcal{W}^c$ be the remaining parameters in the neural network. A loss $L(W, \mathcal{W}^c)$ is *scale-invariant* in $W$ if $L(cW, \mathcal{W}^c) = L(W, \mathcal{W}^c)$ for all $c > 0$; we drop $\mathcal{W}^c$ and use the shorthand $L(W)$ from now on. The norm dynamics below are derived under the noise-dominated model in assumptions 4.1 and 4.2, without assuming scale invariance. We use scale invariance afterward, for the joint parameter group in section 4.2, to interpret effective learning rates.

**Decoupled weight decay and angular motion.** Let $W_t$ be the parameter matrix at step $t$, $u_t$ be the base optimizer update for this matrix, $\eta_t$ be the learning rate, $\lambda$ be the weight decay coefficient, and $\alpha_t := 1 - \eta_t\lambda$. Then the update rule with decoupled weight decay (Loshchilov & Hutter, 2019) is

$$W_{t+1} = \alpha_t W_t - \eta_t u_t. \tag{2}$$

Let $R_t := \|W_t\|_{\mathrm{F}}$ be the radius, $\eta_t^{\mathrm{eff}} := \eta_t \|u_t\|_{\mathrm{F}} / R_t$ the effective learning rate, and $\widehat{W}_t := W_t/R_t$ the matrix direction. The angular step size is the chord distance between the normalized weights $\widehat{W}_t$ and $\widehat{W}_{t+1}$ on the unit Frobenius sphere, $\eta_t^{\mathrm{ang}} := \left\|\widehat{W}_{t+1} - \widehat{W}_t\right\|_{\mathrm{F}}$. Note that $\widehat{W}_{t+1}$ is

$$\widehat{W}_{t+1} = \frac{\alpha_t \widehat{W}_t - \eta_t u_t / R_t}{\left\|\alpha_t \widehat{W}_t - \eta_t u_t / R_t\right\|_{\mathrm{F}}}. \tag{3}$$

Let $u_{t,\perp} := u_t - \left\langle u_t, \widehat{W}_t \right\rangle \widehat{W}_t$ be the tangential component. For $\alpha_t > 0$ and $\eta_t^{\mathrm{eff}}/\alpha_t \ll 1$, (3) gives

$$\eta_t^{\mathrm{ang}} = \frac{\eta_t \|u_{t,\perp}\|_{\mathrm{F}}}{\alpha_t R_t} + O\big((\eta_t^{\mathrm{eff}}/\alpha_t)^2\big). \tag{4}$$

Hence $\eta_t^{\text{ang}} \approx \eta_t^{\text{eff}}$ for mostly tangential updates and $\eta_t\lambda \ll 1$. The effective learning rate includes radial update components and therefore serves as a proxy for the actual angular step size. For a fixed update and weight direction in this regime, increasing $R_t$ suppresses angular motion, so weight decay controls it indirectly through the radius.

**Base optimizers.** The analysis uses idealized AdamW, SVD Muon, and Moonlight scaled SVD Muon as examples. Let $g_t := \nabla_W \ell_t(W_t)$ be the stochastic gradient of the block for the mini-batch loss $\ell_t$, and let $m_t$ (AdamW) and $M_t$ (Muon) be the momentum with coefficient $\beta_1$. For a matrix $A$ with compact singular value decomposition $A = P\Sigma Q^\top$, define the exact SVD matrix sign map $\operatorname{msign}(A) := PQ^\top$. The base update $u_t$ is Adam's preconditioned momentum for AdamW (Loshchilov & Hutter, 2019), and $\operatorname{msign}(M_t)$ scaled by $s_\mu := \max(1, \sqrt{d_{\text{out}}/d_{\text{in}}})$ for Muon (Jordan et al., 2024) or by $s_{\text{moon}} := 0.2\sqrt{\max(d_{\text{in}}, d_{\text{out}})}$ for Moonlight (Liu et al., 2025) (section A.3).

**Idealized stationary model.** We consider constant hyperparameter training, with constant learning rate $\eta$ and constant weight decay $\lambda$, and write $\alpha := 1 - \eta\lambda$ with $0 < \alpha < 1$. Two idealizations make the norm dynamics explicitly computable.

**Assumption 4.1** (Infinite history optimizer)**.** The gradient sequence $\{g_t\}_{t\in\mathbb{Z}}$ and the weight sequence $\{W_t\}_{t\in\mathbb{Z}}$ are defined for all integer times, and the optimizer states are computed from this infinite past, i.e., $m_t = M_t = (1-\beta_1)\sum_{i\geq 0}\beta_1^i g_{t-i}$; $W_t$ is the solution obtained by running (2) from the infinite past. This is the regime where the dependence on the initial optimizer state and weight has decayed.

**Assumption 4.2** (Isotropic stationary gradients and idealized base maps)**.** Let $p = d_{\text{out}}$, $q = d_{\text{in}}$, $d = pq$, and $r = \min(p, q)$. After vectorizing the matrix block, the stochastic optimizer input is an iid isotropic Gaussian sequence, $g_t \sim \mathcal{N}(0, \sigma^2 I_d)$ with $\{g_t\}_{t\in\mathbb{Z}}$ independent. For AdamW, the second moment estimate is replaced by its stationary value $\sigma^2$, so that $u_t = m_t/\sigma$ (section A.4).

Assumption 4.2 is intentionally idealized (it ignores anisotropy, layer specific structure, and the signal in $\mathbb{E}[g_t]$); it provides an easy to compute ansatz for update norms, autocorrelations, radial equilibria, and angular step estimates.

**Update norm and autocorrelation.** For Muon, the correlation between updates at different steps is governed by the SVD sign kernel: for a Gaussian input correlation $\rho \in [-1, 1]$,

$$\kappa_{p,q}(\rho) := \frac{1}{\min(p,q)}\,\mathbb{E}\Big[\Big\langle \operatorname{msign}(X),\, \operatorname{msign}(\rho X + \sqrt{1-\rho^2}\, Z)\Big\rangle\Big], \tag{5}$$

where $X, Z \in \mathbb{R}^{p\times q}$ have iid $\mathcal{N}(0,1)$ entries and are independent. The normalization gives $\kappa_{p,q}(1) = 1$ and $\kappa_{p,q}(0) = 0$, and a Hermite expansion shows $0 \leq \kappa_{p,q}(\rho) \leq \rho$ for $0 \leq \rho \leq 1$ (section A.5).

**Lemma 4.3** (Update norm and update autocorrelation)**.** *Under assumptions 4.1 and 4.2, define the update scale $U$ and the normalized autocorrelation sequence ($c_0 = 1$ and, for every lag $h \geq 1$, $c_h$) by*

$$U = \begin{cases} \sqrt{\frac{1-\beta_1}{1+\beta_1}}\,\sqrt{pq} & \textit{(idealized AdamW)},\\ \sqrt{p} & \textit{(SVD Muon)},\\ 0.2\sqrt{pq} & \textit{(Moonlight)}, \end{cases} \qquad c_h = \begin{cases} \beta_1^h & \textit{(idealized AdamW)},\\ \kappa_{p,q}(\beta_1^h) & \textit{(SVD Muon, Moonlight)}. \end{cases} \tag{6}$$

*Then for every lag $h \geq 1$,*

$$\mathbb{E}\,\|u_t\|_{\text{F}}^2 = U^2, \qquad \mathbb{E}\,\langle u_t,\, u_{t-h}\rangle = U^2 c_h. \tag{7}$$

**Equilibrium radius and angular step estimate.** Under assumptions 4.1 and 4.2, the base optimizer enters the weight decay dynamics only through $U$ and the autocorrelation sum

$$C_\alpha := \sum_{h=1}^{\infty} \alpha^{h-1} c_h, \tag{8}$$

which equals $\beta_1/(1-\alpha\beta_1)$ for idealized AdamW and $\sum_{h\geq 1}\alpha^{h-1}\kappa_{p,q}(\beta_1^h)$ for SVD Muon and Moonlight (finite since $\kappa_{p,q}(\beta_1^h) \leq \beta_1^h$; see section B). In stationarity the current update is negatively correlated with the current weight, $\mathbb{E}\,\langle W_t,\, u_t\rangle = -\eta C_\alpha U^2$ (lemma B.1); substituting this correlation into the second moments of (2) gives the stationary radius and an equilibrium angular step estimate.

| Quantity | AdamW | SVD Muon | Moonlight SVD Muon |
|---|---|---|---|
| Update norm $U$ | $\sqrt{\frac{1-\beta_1}{1+\beta_1}}\sqrt{d_{\text{in}}d_{\text{out}}}$ | $\sqrt{d_{\text{out}}}$ | $0.2\sqrt{d_{\text{in}}d_{\text{out}}}$ |
| Autocorr. sum $C_\alpha$ | $\frac{\beta_1}{1-\alpha\beta_1}$ | $\sum_{h\geq 1}\alpha^{h-1}\kappa_{p,q}(\beta_1^h)$ | $\sum_{h\geq 1}\alpha^{h-1}\kappa_{p,q}(\beta_1^h)$ |
| Equilibrium norm $R_\star$ | | $\eta U\sqrt{\frac{1+2\alpha C_\alpha}{1-\alpha^2}}$ | |
| Cosine proxy $\cos_\star$ | | $-C_\alpha\sqrt{\frac{1-\alpha^2}{1+2\alpha C_\alpha}}$ | |
| Angular step estimate $(\eta_{\text{eq}}^{\text{ang}})^2$ | | $\frac{2(1-\alpha)\big(1-(1-\alpha)C_\alpha\big)}{1+2\alpha C_\alpha}$ | |

Table 1: Steady state ansatz quantities for $\alpha := 1-\eta\lambda$, with $C_\alpha$ from (8) and the cosine proxy $\cos_\star := \mathbb{E}\langle W_t, u_t\rangle/(R_\star U)$ derived in corollary B.2. SVD Muon and Moonlight use the matrix sign autocorrelation kernel $\kappa_{p,q}$ of (5), so their equilibrium angular step estimates need not equal AdamW's.

**Theorem 4.4** (Weight decay sets equilibrium radius and angular step estimate)**.** *Under assumptions 4.1 and 4.2 and constant hyperparameter training with $\alpha = 1-\eta\lambda$, the second moment $\mathbb{E}\|W_t\|_{\text{F}}^2$ of the weight in the decoupled weight decay update (2) converges to the squared equilibrium radius $R_\star^2$, with the equilibrium angular step estimate $\eta_{\text{eq}}^{\text{ang}} := \sqrt{\mathbb{E}\|W_{t+1}-W_t\|_{\text{F}}^2}/R_\star$ given by*

$$R_\star = \eta U\sqrt{\frac{1+2\alpha C_\alpha}{1-\alpha^2}}, \qquad (\eta_{\text{eq}}^{\text{ang}})^2 = \frac{2(1-\alpha)\big(1-(1-\alpha)C_\alpha\big)}{1+2\alpha C_\alpha}. \tag{9}$$

*For idealized AdamW, $C_\alpha = \beta_1/(1-\alpha\beta_1)$ and*

$$\eta_{\text{eq}}^{\text{ang}} = \sqrt{\frac{2(1-\alpha)(1-\beta_1)}{1+\alpha\beta_1}} = \sqrt{\frac{2\eta\lambda(1-\beta_1)}{1+(1-\eta\lambda)\beta_1}}. \tag{10}$$

This stationary moment proxy estimates the root mean square angular step size when radii concentrate near $R_\star$; it differs from the equilibrium estimate $\eta U/R_\star$ of the effective learning rate. For small $\eta\lambda$, the AdamW expression gives $\eta_{\text{eq}}^{\text{ang}} \approx \sqrt{2(1-\beta_1)/(1+\beta_1)}\sqrt{\eta\lambda}$. In all cases, when $\eta\lambda$ is small and $\beta_1$ is fixed, $R_\star = \Theta(U\sqrt{\eta/\lambda})$ up to the autocorrelation factor $\sqrt{1+2\alpha C_\alpha}$ (section B gives the AdamW closed form). Closely related estimates for AdamW update and weight RMS based on mean field approximation appear in Su (2025b;c;d). Table 1 consolidates the stationary quantities.

**Group scale invariance.** Under the assumptions above, theorem 4.4 predicts each matrix's equilibrium radius $R_\star$ and a constant equilibrium angular step estimate. Let $G$ collect the input embedding and every attention and MLP output projection, with the remaining parameters fixed. In a bias free prenorm Transformer with a final normalization and an untied output head, jointly scaling these matrices by $c > 0$ leaves the loss unchanged, $L(cG) = L(G)$ (fig. 12).[1] Scaling the embedding multiplies the initial residual stream by $c$. Each branch normalizes its input, so scaling its output projection multiplies its residual contribution by the same $c$. The final normalization removes this common factor. Thus the loss depends only on the direction $\widehat{G}_t := G_t/\|G_t\|_2$, where $G_t$ concatenates the matrices $W_{i,t}$ in $G$ at step $t$ into one vector. We measure the group's movement by its angular step size $\eta_{G,t}^{\text{ang}} := \left\|\widehat{G}_{t+1} - \widehat{G}_t\right\|_2$. Let $R_{i,t} := \|W_{i,t}\|_{\text{F}}$ and $\eta_{i,t}^{\text{ang}}$ be the radius and angular step size of each matrix. If the ratios between these radii remain fixed over the step, the squared group angular step is the weighted average $(\eta_{G,t}^{\text{ang}})^2 = \sum_i R_{i,t}^2(\eta_{i,t}^{\text{ang}})^2/\sum_i R_{i,t}^2$. For pretraining configurations with shared optimizer hyperparameters, theorem 4.4 motivates approximating every $\eta_{i,t}^{\text{ang}}$ by a common value $\bar{\eta}^{\text{ang}}$. The weighted average then gives $\eta_{G,t}^{\text{ang}} \approx \bar{\eta}^{\text{ang}}$. For small, mostly tangential updates with $\eta_t\lambda \ll 1$, each matrix's effective learning rate $\eta_t^{\text{eff}}$ approximates its angular step size by (4), and hence also approximates $\eta_{G,t}^{\text{ang}}$. Hyperball sets the effective learning rate of each

[1] This idealization omits normalization epsilon and assumes there are no unscaled additive inputs to the residual stream. With tied embeddings, preserving the logits also requires a compensating inverse scaling of the final normalization gain.

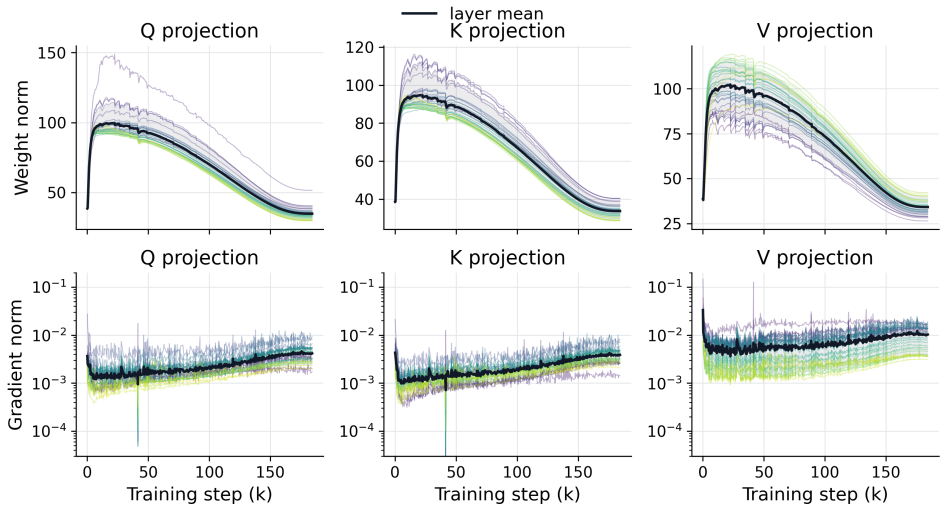


Figure 6: Weight norm and gradient norm diagnostics for a 1.2B AdamW run (Phenomena 1 and 2 in section 4.3). Top: Q/K/V projection weight norms across all 32 layers. Bottom: the corresponding gradient norms. Thin curves are individual layers. The dark curve is the layer mean. Weight norms follow the learning rate schedule, and gradient norms rise as weight norms shrink during decay.

constrained matrix directly to $\eta_t$ given this reasoning. Scale invariance also makes the gradient scale inversely with the weights.

**Lemma 4.5** (Inverse gradient scaling for scale-invariant losses)**.** *If $L : \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}} \to \mathbb{R}$ is differentiable and scale-invariant, $L(cW) = L(W)$ for all $c > 0$, then $\nabla_W L(cW) = \frac{1}{c} \nabla_W L(W)$ for all $c > 0$.*

For a scale-invariant block $W$ (such as the concatenated group $G$), applying lemma 4.5 to $W_t = R_t \widehat{W}_t$ gives $\|\nabla_W L(W_t)\|_{\text{F}} = \left\|\nabla_W L(\widehat{W}_t)\right\|_{\text{F}} / R_t$. When the block radius also follows $R_\star \propto \sqrt{\eta/\lambda}$ as in (9), this gives the scale-invariant prediction $\|\nabla_W L(W_t)\|_{\text{F}} \propto \sqrt{\lambda/\eta_t}$ when the direction of the weight is fixed and the autocorrelation factor changes slowly.

### 4.3 Empirical Validation

**Phenomenon 1: weight norm tracks learning rate warmup and decay throughout training.** Under a WSD learning rate schedule, (9) predicts that $R_t$ should rise during warmup and shrink during learning rate decay. In a 1.2B AdamW run with cosine learning rate decay, Q/K/V projection norms across layers show exactly this pattern: norms rise rapidly during warmup and then decrease during decay (fig. 6 in section C, top).

**Phenomenon 2: gradient norm increases through training.** For scale-invariant blocks, lemma 4.5 predicts that gradient norms scale approximately as $1/R_t$. This phenomenon is studied in Defazio (2025), where a similar explanation is provided. In the same run, the corresponding Q/K/V gradient norms increase during training as the weight norms shrink during learning rate decay (fig. 6, bottom).

**Phenomenon 3: when $\eta\lambda$ is fixed, AdamW converges to essentially the same loss while each matrix norm is roughly proportional to $\eta$.** Theorem 4.4 predicts that holding $\eta\lambda$ fixed keeps the equilibrium angular step estimate $\eta_{\text{eq}}^{\text{ang}}$ fixed, so the training loss should be nearly unchanged. Furthermore, if we divide $\lambda$ by $c$ and multiply $\eta$ by $c$, then (9) predicts $R_\star \propto c$. In fig. 7, we present two runs with $(\eta, \lambda) = (0.002, 0.2)$ and $(0.004, 0.1)$, and we observe that the train loss curves nearly overlap, while the equilibrium Q/K norms are roughly doubled in the larger learning rate run.

**Phenomenon 4: despite sharing the same learning rate schedule, weight decay starts with a higher loss but ultimately converges lower than no weight decay.** When the WD and no WD runs use the same learning rate warmup and decay schedule, (9) and (10) predict different weight norms and angular step dynamics. Without WD, the weight norm grows and the effective learning rate $\eta_t^{\text{eff}}$ decays; with WD, the run maintains a larger effective learning rate throughout training. Under the small-step and tangential approximation in section 4.2, this indicates a larger angular step size. In

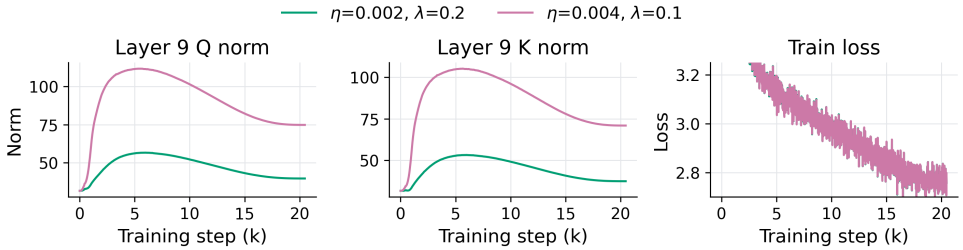


Figure 7: Ablation with fixed $\eta\lambda$. Two AdamW runs with the same product $\eta\lambda = 4 \cdot 10^{-4}$ have nearly identical train loss, while the larger learning rate run has roughly doubled layer 9 Q/K norms.

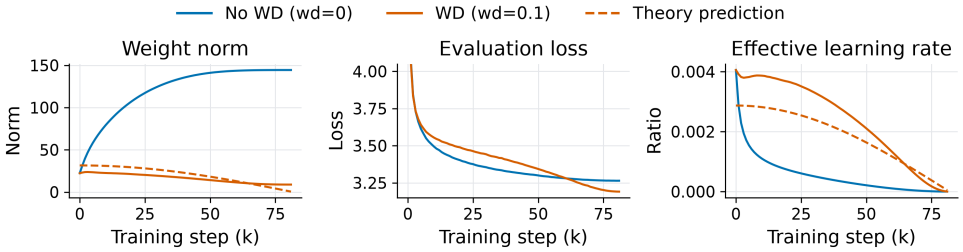


Figure 8: Weight decay versus no weight decay under the same learning rate schedule: without WD the weight norm grows and the effective learning rate $\eta_t^{\text{eff}}$ decays, with WD the run keeps a larger late effective learning rate and crosses over to a lower validation loss. Dashed: theory prediction for the WD effective learning rate.

the river valley picture (Wen et al., 2024), the loss decomposes into a relatively flat "river" direction along which long term progress happens and steep "hill" directions excited by stochastic gradients. A larger angular step size amplifies the hill oscillations, which raises the observed loss early in training, but also accelerates motion along the river; when the learning rate decays, the oscillations shrink and the iterate settles closer to the riverbed, revealing the progress already made along the river. Hence the WD run starts with a higher loss but ultimately reaches a lower loss (fig. 8).

**Phenomenon 5: contrary to the original $\mu$P prediction, transfer is not sensitive to weight scale at initialization but is sensitive to weight decay scaling.** Recent hyperparameter transfer studies find that transfer is often less sensitive to the initial weight scale than to how weight decay is scaled across model size and training duration (Kosson et al., 2025; Blake et al., 2024; Fan et al., 2025; Wang & Aitchison, 2024; Qiu et al., 2025). This is consistent with (9): the radial recursion forgets the initial radius and converges to a norm set by $\eta$, $\lambda$, and the optimizer dependent update norm $U$. Changing the scaling rule for $\lambda$, however, changes both the equilibrium norm $R_\star$ and the equilibrium angular step estimate in (10), so it changes the dynamics relevant to transfer that are assumed by $\mu$P style analyses (Yang et al., 2022). Hyperball turns this dependence into an explicit design choice by fixing the radius and normalized update length directly, which is why the same learning rate window transfers better across depths and widths in Section 3.3.

## 5 Related Work

**Fixed norms in LLM pretraining.** Several recent works enforce normalization for language model pretraining: nGPT (Loshchilov et al., 2024) enforces columnwise unit norms, Nemotron-Flash (Fu et al., 2025) applies per channel spherical constraints to weights but not updates, anGPT (Franke et al., 2025) bounds each weight row using constrained parameter regularization (Franke et al., 2024), and Owen et al. (2025) periodically rescale weights toward a target variance. On richer manifolds, Modular Manifolds (Bernstein, 2025a), Muon+Stiefel (Su, 2025a), orthogonal manifold steepest descent (Bernstein, 2025b; Cesista, 2025), Newhouse et al. (2025), and SSO (Xie et al., 2026) optimize on the Stiefel or spectral sphere manifold, and related spectral norm views of Muon and weight decay appear in Su (2024); Chen et al. (2025). Hyperball projects onto the matrix Frobenius sphere, a softer constraint than normalization by column or channel, at $O(N^2)$ cost per matrix versus $O(N^3)$ for spectral projections. We also explicitly control the update norm and the effective learning rate. Additional related work appears in section D.

### AI USE STATEMENT

In this work, we used generative AI tools for formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, and implementing methods (parts of the Hyperball optimizer implementation and of the training and evaluation code). We have not used generative AI tools for developing the theoretical model or conceptual framework, proposing or refining hypotheses, designing or providing feedback on the research methodology or experiments, or interpreting results; generating synthetic datasets, translation, cleaning or reformatting datasets, and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for creating and editing the code that produces the figures, for drafting parts of the manuscript, and for editing the manuscript to improve readability and to fit the page limit, including reorganizing material between the main text and the appendix. We have reviewed all AI-assisted work: every proof step suggested by an AI tool was checked by the authors, and the proofs in section B were verified by the authors line by line; AI-generated code was reviewed and tested by the authors, and all reported numbers come from experiments run and logged by the authors; figures were checked against the underlying experiment logs; all AI-drafted or AI-edited text was reviewed and revised by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

### REPRODUCIBILITY STATEMENT

The Hyperball update is fully specified in eq. (1) and algorithm 1, together with the choice of which parameter matrices are constrained (section 2). The theoretical results in section 4 state their assumptions explicitly (assumptions 4.1 and 4.2), and all proofs are given in section B. The architecture, data mixture, baselines, speedup metric, and learning rate transfer protocol used in the experiments are described in section 3.1, and the public modded-nanogpt Track 3 entries are referenced in section 3.2.

## REFERENCES


Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*, 2023. URL https://arxiv.org/abs/2310.10631.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. In *NeurIPS Deep Learning Symposium*, 2016. URL https://arxiv.org/abs/1607.06450.

Jeremy Bernstein. Modular manifolds. https://thinkingmachines.ai/blog/modular-manifolds/, 2025a.

Jeremy Bernstein. Orthogonal manifold. https://docs.modula.systems/algorithms/manifold/orthogonal/, 2025b.

Charlie Blake, Constantin Eichenberg, Josef Dean, Lukas Balles, Luke Y. Prince, Björn Deiseroth, Andres Felipe Cruz-Salinas, Carlo Luschi, Samuel Weinbach, and Douglas Orr. u-$\mu$P: The unit-scaled maximal update parametrization. *arXiv preprint arXiv:2407.17465*, 2024. URL https://arxiv.org/abs/2407.17465.

Franz Louis Cesista. Heuristic solutions for steepest descent on the stiefel manifold. https://leloykun.github.io/ponder/steepest-descent-stiefel/, 2025.

Lizhang Chen, Jonathan Li, and Qiang Liu. Muon optimizes under spectral norm constraints. *arXiv preprint arXiv:2506.15054*, 2025. URL https://arxiv.org/abs/2506.15054.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms. In *NeurIPS*, 2023. URL https://arxiv.org/abs/2302.06675.

Francesco D'Angelo, Maksym Andriushchenko, Aditya Varre, and Nicolas Flammarion. Why do we need weight decay in modern deep learning? *arXiv preprint arXiv:2310.04415*, 2023. URL https://arxiv.org/abs/2310.04415.

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

Zhiyuan Fan, Yifeng Liu, Qingyue Zhao, Angela Yuan, and Quanquan Gu. Robust layerwise scaling rules by proper weight decay tuning. *arXiv preprint arXiv:2510.15262*, 2025. URL https://arxiv.org/abs/2510.15262.

Jörg K. H. Franke, Michael Hefenbrock, Gregor Koehler, and Frank Hutter. Improving deep learning optimization through constrained parameter regularization. In *NeurIPS*, 2024. URL https://arxiv.org/abs/2311.09058.

Jörg K. H. Franke, Urs Spiegelhalter, Marianna Nezhurina, Jenia Jitsev, Frank Hutter, and Michael Hefenbrock. Learning in compact spaces with approximately normalized transformer. In *NeurIPS*, 2025. URL https://arxiv.org/abs/2505.22014.

Yonggan Fu, Xin Dong, Shizhe Diao, Matthijs Van keirsbilck, Hanrong Ye, Wonmin Byeon, Yashaswi Karnati, Lucas Liebenwein, Hannah Zhang, Nikolaus Binder, Maksim Khadkevich, Alexander Keller, Jan Kautz, Yingyan Celine Lin, and Pavlo Molchanov. Nemotron-flash: Towards latency-optimal hybrid small language models. In *NeurIPS*, 2025. URL https://arxiv.org/abs/2511.18890.

Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. Query-key normalization for transformers. In *Findings of EMNLP*, pp. 4246–4253, 2020. doi: 10.18653/v1/2020.findings-emnlp.379. URL https://aclanthology.org/2020.findings-emnlp.379/.

Byeongho Heo, Sanghyuk Chun, Seong Joon Oh, Dongyoon Han, Sangdoo Yun, Gyuwan Kim, Youngjung Uh, and Jung-Woo Ha. AdamP: Slowing down the slowdown for momentum optimizers on scale-invariant weights. In *ICLR*, 2021. URL https://arxiv.org/abs/2006.08217.

Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. Norm matters: Efficient and accurate normalization schemes in deep networks. In *NeurIPS*, 2018. URL https://arxiv.org/abs/1803.01814.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022. URL https://arxiv.org/abs/2203.15556.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. URL https://arxiv.org/abs/1502.03167.

Keller Jordan. Modded-NanoGPT optimization benchmark: Track 3 optimization. https://github.com/KellerJordan/modded-nanogpt/tree/master/records/track_3_optimization, 2026. Accessed 2026-05-17.

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

Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. *arXiv preprint arXiv:2312.02696*, 2023. URL https://arxiv.org/abs/2312.02696.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. URL https://arxiv.org/abs/1412.6980.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (BiT): General visual representation learning. In *ECCV*, 2020. URL https://arxiv.org/abs/1912.11370.

Atli Kosson, Bettina Messmer, and Martin Jaggi. Rotational equilibrium: How weight decay balances learning across neural networks. In *ICML*, 2024a. URL https://arxiv.org/abs/2305.17212.

Atli Kosson, Bettina Messmer, and Martin Jaggi. Analyzing and reducing the need for learning rate warmup in GPT training. In *NeurIPS*, 2024b. URL https://arxiv.org/abs/2410.23922.

Atli Kosson, Jeremy Welborn, Yang Liu, Martin Jaggi, and Xi Chen. Weight decay may matter more than $\mu$P for learning rate transfer in practice. *arXiv preprint arXiv:2510.19093*, 2025. URL https://arxiv.org/abs/2510.19093.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, et al. DataComp-LM: In search of the next generation of training sets for language models. In *NeurIPS Datasets and Benchmarks*, 2024. URL https://arxiv.org/abs/2406.11794.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. StarCoder: May the source be with you! *arXiv preprint arXiv:2305.06161*, 2023. URL https://arxiv.org/abs/2305.06161.

Zhiyuan Li, Kaifeng Lyu, and Sanjeev Arora. Reconciling modern deep learning with traditional optimization analyses: The intrinsic learning rate. In *NeurIPS*, 2020. URL https://arxiv.org/abs/2010.02916.

Zichong Li, Liming Liu, Chen Liang, Weizhu Chen, and Tuo Zhao. NorMuon: Making Muon more efficient and scalable. *arXiv preprint arXiv:2510.05491*, 2025. URL https://arxiv.org/abs/2510.05491.

Wu Lin, Scott C. Lowe, Felix Dangel, Runa Eschenhagen, Zikun Xu, and Roger B. Grosse. Understanding and improving shampoo and SOAP via Kullback-Leibler minimization. In *ICLR*, 2026. URL https://arxiv.org/abs/2509.03378.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, et al. Muon is scalable for LLM training. *arXiv preprint arXiv:2502.16982*, 2025. URL https://arxiv.org/abs/2502.16982.

Sheng Liu, Xiao Li, Yuexiang Zhai, Chong You, Zhihui Zhu, Carlos Fernandez-Granda, and Qing Qu. Convolutional normalization: Improving deep convolutional network robustness and training. In *NeurIPS*, 2021. URL https://arxiv.org/abs/2103.00673.

Weiyang Liu, Zhen Liu, Zhiding Yu, Bo Dai, Rongmei Lin, Yisen Wang, James M. Rehg, and Le Song. Decoupled networks. In *CVPR*, 2018. URL https://arxiv.org/abs/1804.08071.

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. *arXiv preprint arXiv:2608.24814*, 2026. URL https://arxiv.org/abs/2608.24814.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. URL https://arxiv.org/abs/1711.05101.

Ilya Loshchilov, Cheng-Ping Hsieh, Simeng Sun, and Boris Ginsburg. nGPT: Normalized transformer with representation learning on the hypersphere. *arXiv preprint arXiv:2410.01131*, 2024. URL https://arxiv.org/abs/2410.01131.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018. URL https://arxiv.org/abs/1802.05957.

Takeru Miyato, Sindy Löwe, Andreas Geiger, and Max Welling. Artificial Kuramoto oscillatory neurons. In *ICLR*, 2025. URL https://arxiv.org/abs/2410.13821.

Laker Newhouse, R. Preston Hess, Franz Cesista, Andrii Zahorodnii, Jeremy Bernstein, and Phillip Isola. Training transformers with enforced Lipschitz constants. *arXiv preprint arXiv:2507.13338*, 2025. URL https://arxiv.org/abs/2507.13338.

Louis Owen, Abhay Kumar, Nilabhra Roy Chowdhury, and Fabian Güra. Variance control via weight rescaling in LLM pre-training. *arXiv preprint arXiv:2503.17500*, 2025. URL https://arxiv.org/abs/2503.17500.

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 *NeurIPS Datasets and Benchmarks*, 2024. URL https://arxiv.org/abs/2406.17557.

Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. Micro-batch training with batch-channel normalization and weight standardization. *arXiv preprint arXiv:1903.10520*, 2019. URL https://arxiv.org/abs/1903.10520.

Shikai Qiu, Zixi Chen, Hoang Phan, Qi Lei, and Andrew Gordon Wilson. Hyperparameter transfer enables consistent gains of matrix-preconditioned optimizers across scales. In *NeurIPS*, 2025. URL https://arxiv.org/abs/2512.05620.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL https://www.jmlr.org/papers/v21/20-074.html.

Simon Roburin, Yann de Mont-Marin, Andrei Bursuc, Renaud Marlet, Patrick Pérez, and Mathieu Aubry. Spherical perspective on learning with normalization layers. *arXiv preprint arXiv:2006.13382*, 2020. URL https://arxiv.org/abs/2006.13382.

Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NeurIPS*, 2016. URL https://arxiv.org/abs/1602.07868.

Jianlin Su. Thinking about spectral norm gradient and spectral weight decay. https://kexue.fm/archives/10648, 2024.

Jianlin Su. Muon on the stiefel manifold. https://kexue.fm/archives/11221, 2025a.

Jianlin Su. Why Adam's update RMS is 0.2? https://kexue.fm/archives/11267, 2025b.

Jianlin Su. AdamW weight RMS asymptotics (part I). https://kexue.fm/archives/11307, 2025c.

Jianlin Su. AdamW weight RMS asymptotics (part II). https://kexue.fm/archives/11404, 2025d.

Twan van Laarhoven. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*, 2017. URL https://arxiv.org/abs/1706.05350.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. URL https://arxiv.org/abs/1706.03762.

Ruosi Wan, Zhanxing Zhu, Xiangyu Zhang, and Jian Sun. Spherical motion dynamics: Learning dynamics of normalized neural network using SGD and weight decay. In *Advances in Neural Information Processing Systems*, volume 34, pp. 6380–6391, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/326a8c055c0d04f5b06544665d8bb3ea-Paper.pdf.

Xi Wang and Laurence Aitchison. How to set AdamW's weight decay as you scale model and dataset size. *arXiv preprint arXiv:2405.13698*, 2024. URL https://arxiv.org/abs/2405.13698.

Kaiyue Wen, Zhiyuan Li, Jason Wang, David Hall, Percy Liang, and Tengyu Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape perspective. *arXiv preprint arXiv:2410.05192*, 2024. URL https://arxiv.org/abs/2410.05192.

Kaiyue Wen, David Hall, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them. *arXiv preprint arXiv:2509.02046*, 2025. URL https://arxiv.org/abs/2509.02046.

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

Tian Xie, Haoming Luo, Haoyu Tang, Yiwen Hu, Jason Klein Liu, Qingnan Ren, Yang Wang, Wayne Xin Zhao, Rui Yan, Bing Su, Chong Luo, and Baining Guo. Controlled LLM training on spectral sphere. *arXiv preprint arXiv:2601.08393*, 2026. URL https://arxiv.org/abs/2601.08393.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In *ICML*, 2020. URL https://arxiv.org/abs/2002.04745.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs V: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv preprint arXiv:2203.03466*, 2022. URL https://arxiv.org/abs/2203.03466.

Greg Yang, James B. Simon, and Jeremy Bernstein. A spectral condition for feature learning. *arXiv preprint arXiv:2310.17813*, 2023. URL https://arxiv.org/abs/2310.17813.

Jinghui Yuan, Hongtao Zhang, Jade Zou, Tianyu Li, Wenjie Zhou, Tianyu He, and Wei Chen. HyperTransfer: Understanding the equivalence between base optimizer and Hyperball. *arXiv preprint arXiv:2609.07017*, 2026. URL https://arxiv.org/abs/2609.07017.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *NeurIPS*, 2019. URL https://arxiv.org/abs/1910.07467.

Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger Grosse. Three mechanisms of weight decay regularization. In *ICLR*, 2019. URL https://openreview.net/forum?id=B1lz-3Rct7.

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.

Zhijian Zhuo, Yutao Zeng, Ya Wang, Sijun Zhang, Jian Yang, Xiaoqing Li, Xun Zhou, and Jinwen Ma. HybridNorm: Towards stable and efficient transformer training via hybrid normalization. In *NeurIPS*, 2025. URL https://arxiv.org/abs/2503.04598.

# A Additional details on Hyperball and the analysis

## A.1 Exact displacement of the Hyperball step

Write $\widehat{u}_t := u_t / \|u_t\|_{\mathrm{F}}$ and $\widehat{W}_t := W_t / \|W_t\|_{\mathrm{F}}$. The Hyperball update (1) has the exact displacement

$$W_{t+1} - W_t = R \left( \frac{\widehat{W}_t - \eta_t \widehat{u}_t}{\left\| \widehat{W}_t - \eta_t \widehat{u}_t \right\|_{\mathrm{F}}} - \widehat{W}_t \right), \tag{11}$$

so radial renormalization returns the trial point to the sphere of radius $R$. Consequently, the angular step size is $\eta_t^{\mathrm{ang}} = \|W_{t+1} - W_t\|_{\mathrm{F}} / R$, while the effective learning rate is $\eta_t$. For small $\eta_t$, $\eta_t^{\mathrm{ang}} = \eta_t \sqrt{1 - \left\langle \widehat{u}_t, \widehat{W}_t \right\rangle^2} + O(\eta_t^2)$, which is approximately $\eta_t$ for mostly tangential updates.

## A.2 Frobenius versus spectral constraint

A natural alternative to the Frobenius norm constraint is a spectral norm constraint. Let $\|W\|_{\mathrm{op}}$ denote the operator norm of a matrix. The spectral condition of Yang et al. (2023) identifies relative spectral update size as a central quantity for feature learning, and SSO constrains training to the spectral sphere by steepest descent or projection (Xie et al., 2026). In this view, a spectral Hyperball variant would fix $\|W_t\|_{\mathrm{op}}$ and normalize the update in operator norm, directly controlling the sharpest layerwise scaling factor.

We use the Frobenius version in (1) for two reasons: computation cost and the theoretical motivation in section 4. Projection onto the Frobenius sphere is $O(N^2)$ per matrix, whereas exact spectral projection generally requires an SVD and costs $O(N^3)$. One diagnostic for whether Frobenius control is close to spectral control is the stable rank ratio. For a parameter matrix $W \in \mathbb{R}^{d_{\mathrm{out}} \times d_{\mathrm{in}}}$, define

$$\mathcal{R}(W) := \frac{\|W\|_{\mathrm{F}}^2}{\|W\|_{\mathrm{op}}^2} \in [1, \min(d_{\mathrm{in}}, d_{\mathrm{out}})]. \tag{12}$$

Values satisfying $\mathcal{R}(W) = \Omega(\min(d_{\mathrm{in}}, d_{\mathrm{out}}))$ indicate that the singular value spectrum is not dominated by a single direction, in which case Frobenius constraints behave similarly to spectral constraints up to a slowly varying factor. The same high stable rank regime is observed empirically in the Kimi Moonlight analysis (Liu et al., 2025, Appendix F). A spectral Hyperball variant is a natural direction when singular value concentration makes Frobenius control too loose.

## A.3 Concrete base updates

Let $\ell_t$ be the minibatch loss, $g_t := \nabla_W \ell_t(W_t)$ be the stochastic gradient for the matrix block, let $\epsilon > 0$ be Adam's numerical stability constant, and let divisions and square roots in Adam be elementwise. For AdamW (Loshchilov & Hutter, 2019),

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \qquad v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^{\odot 2}, \qquad u_t = \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t} + \epsilon}, \tag{13}$$

where $\widehat{m}_t$ and $\widehat{v}_t$ denote the bias corrected moments. For Muon (Jordan et al., 2024; Liu et al., 2025), let

$$M_t = \beta_1 M_{t-1} + (1 - \beta_1) g_t. \tag{14}$$

For a matrix $A$ with compact singular value decomposition $A = P \Sigma Q^\top$, the exact SVD matrix sign map is

$$\mathrm{msign}(A) := P Q^\top. \tag{15}$$

Muon implementations often compute this map by Newton–Schulz iteration; in this theory we analyze the idealized SVD Muon update. For $W \in \mathbb{R}^{d_{\mathrm{out}} \times d_{\mathrm{in}}}$, define $s_\mu := \max(1, \sqrt{d_{\mathrm{out}} / d_{\mathrm{in}}})$. The Muon base update is

$$u_t = s_\mu \, \mathrm{msign}(M_t). \tag{16}$$

Moonlight (Liu et al., 2025) uses the same momentum and exact SVD sign map, but replaces $s_\mu$ with $s_{\mathrm{moon}} := 0.2 \sqrt{\max(d_{\mathrm{in}}, d_{\mathrm{out}})}$:

$$u_t = s_{\mathrm{moon}} \, \mathrm{msign}(M_t). \tag{17}$$

### A.4 IDEALIZED ADAMW UPDATE

Under assumptions 4.1 and 4.2, for a scalar coordinate $\bar{g}_t$ with $\mathbb{E}[\bar{g}_t^2] = \sigma^2$, the stationary Adam second moment satisfies

$$\mathbb{E}[v_t] = (1-\beta_2)\sum_{i\geq 0}\beta_2^i\,\mathbb{E}[\bar{g}_{t-i}^2] = (1-\beta_2)\sum_{i\geq 0}\beta_2^i\,\sigma^2 = \sigma^2. \tag{18}$$

We therefore replace the Adam denominator by $\sigma$ and ignore the vanishing bias correction transient:

$$u_t = \frac{m_t}{\sigma}, \qquad m_t = (1-\beta_1)\sum_{i\geq 0}\beta_1^i g_{t-i}. \tag{19}$$

### A.5 RANGE OF THE MUON SIGN KERNEL

Let $F(X) := r^{-1/2}\,\mathrm{msign}(X)$ with $r = \min(p,q)$. Since $F(-X) = -F(X)$ and the distribution of $X$ is symmetric, $\mathbb{E}[F(X)] = 0$ and $\mathbb{E}\,\|F(X)\|_{\mathrm{F}}^2 = 1$. By the Hermite expansion of the Gaussian noise operator,

$$\kappa_{p,q}(\rho) = \sum_{k\geq 1} a_k \rho^k, \qquad a_k \geq 0, \qquad \sum_{k\geq 1} a_k = 1. \tag{20}$$

Therefore, for $0 \leq \rho \leq 1$,

$$0 \leq \kappa_{p,q}(\rho) \leq \rho. \tag{21}$$

This inequality bounds the correlation between weight and update, and it makes the series defining $C_\alpha$ in (8) finite for SVD Muon and Moonlight. Figure 9 illustrates this range in a numerical simulation with $p = q = 100$.

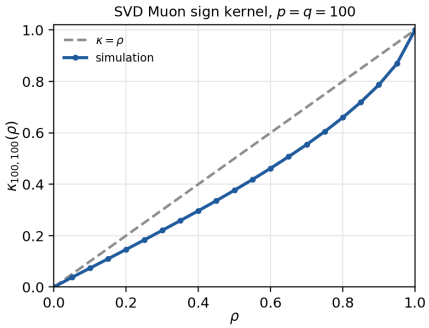


Figure 9: Monte Carlo estimate of $\kappa_{p,q}(\rho)$ for $p = q = 100$.

## B PROOFS

### B.1 PROOF OF LEMMA 4.3

**Lemma 4.3** (Update norm and update autocorrelation)**.** *Under assumptions 4.1 and 4.2, define the update scale $U$ and the normalized autocorrelation sequence ($c_0 = 1$ and, for every lag $h \geq 1$, $c_h$) by*

$$U = \begin{cases} \sqrt{\frac{1-\beta_1}{1+\beta_1}}\,\sqrt{pq} & \textit{(idealized AdamW)},\\ \sqrt{p} & \textit{(SVD Muon)},\\ 0.2\sqrt{pq} & \textit{(Moonlight)}, \end{cases} \qquad c_h = \begin{cases} \beta_1^h & \textit{(idealized AdamW)},\\ \kappa_{p,q}(\beta_1^h) & \textit{(SVD Muon, Moonlight)}. \end{cases} \tag{6}$$

*Then for every lag $h \geq 1$,*

$$\mathbb{E}\,\|u_t\|_{\mathrm{F}}^2 = U^2, \qquad \mathbb{E}\,\langle u_t,\, u_{t-h}\rangle = U^2 c_h. \tag{7}$$

*Proof.* For AdamW, each coordinate of (19) is a stationary Gaussian moving average. If $\bar{g}_t$ is one coordinate, then

$$\bar{u}_t = (1-\beta_1)\sum_{i\geq 0}\beta_1^i\,\frac{\bar{g}_{t-i}}{\sigma}. \tag{22}$$

Thus $\mathbb{E}[\bar{u}_t^2] = (1-\beta_1)/(1+\beta_1)$. For lag $h \geq 1$,

$$\mathbb{E}[\bar{u}_t\bar{u}_{t-h}] = \frac{1-\beta_1}{1+\beta_1}\,\beta_1^h. \tag{23}$$

Summing over $d = pq$ independent coordinates gives the AdamW line of (6) and both identities in (7), with $c_h = \beta_1^h$.

For SVD Muon, $\mathrm{msign}(A)$ has exactly $r = \min(p, q)$ nonzero singular values, all equal to 1. Therefore

$$\|s_\mu \,\mathrm{msign}(M_t)\|_{\mathrm{F}}^2 = s_\mu^2 r = \max(1, p/q)\min(p, q) = p, \tag{24}$$

which gives $U = \sqrt{p}$. For Moonlight, the same calculation gives

$$\|s_{\mathrm{moon}} \,\mathrm{msign}(M_t)\|_{\mathrm{F}}^2 = 0.04 \max(p, q)\min(p, q) = 0.04pq, \tag{25}$$

so $U = 0.2\sqrt{pq}$.

It remains to identify the autocorrelation. The stationary momentum matrices satisfy

$$M_t = (1 - \beta_1) \sum_{i \geq 0} \beta_1^i g_{t-i}. \tag{26}$$

Hence each pair $(M_t, M_{t-h})$ is jointly Gaussian, with identical marginal covariance and entrywise correlation $\beta_1^h$ using standard property of geometric sequences. After dividing both matrices by their common standard deviation, the pair has the same distribution as

$$\left(X,\ \beta_1^h X + \sqrt{1 - \beta_1^{2h}}\, Z\right), \tag{27}$$

with $X, Z$ as in (5). Because the matrix sign is invariant to positive scalar rescaling, the normalized expected inner product of the two SVD Muon updates is exactly $\kappa_{p,q}(\beta_1^h)$. Multiplying by $s_\mu^2 r = U^2$ or $s_{\mathrm{moon}}^2 r = U^2$ gives the autocorrelation identity in (7). The scalar multiplier $s_\mu$ or $s_{\mathrm{moon}}$ cancels in the normalized autocorrelation, so Muon and Moonlight share the same $c_h$. □

### B.2 Stationary projection coefficient

**Lemma B.1** (Stationary projection coefficient)**.** *Under assumptions 4.1 and 4.2 and constant hyperparameter training, let the normalized update autocorrelation sequence $c_h$ be given by (6) and let $C_\alpha$ be defined by (8). Define $\gamma_t := \mathbb{E}\langle W_t,\, u_t\rangle / U^2$ to be the projection coefficient, quantifying how strongly the weight and update correlate. Then $\gamma_t$ is identically equal to a constant value $\gamma$ for all $t$, with*

$$\gamma = -\eta C_\alpha. \tag{28}$$

*Proof.* Since $0 < \alpha < 1$, the stationary solution of the decoupled weight decay recursion (2) is

$$W_t = -\eta \sum_{h=1}^{\infty} \alpha^{h-1} u_{t-h}. \tag{29}$$

Taking the expectation of the inner product with $u_t$ and using (7) gives

$$\mathbb{E}\langle W_t,\, u_t\rangle = -\eta \sum_{h=1}^{\infty} \alpha^{h-1} \mathbb{E}\langle u_{t-h},\, u_t\rangle = -\eta U^2 \sum_{h=1}^{\infty} \alpha^{h-1} c_h. \tag{30}$$

Dividing by $U^2$ and using the definition of $C_\alpha$ in (8) proves (28). □

For AdamW, $c_h = \beta_1^h$, and the sum can be simplified,

$$C_\alpha^{\mathrm{Adam}} = \sum_{h=1}^{\infty} \alpha^{h-1}\beta_1^h = \beta_1 \sum_{h=1}^{\infty} (\alpha\beta_1)^{h-1} = \frac{\beta_1}{1 - \alpha\beta_1}. \tag{31}$$

For SVD Muon and Moonlight,

$$C_\alpha^{\mathrm{Muon}} = C_\alpha^{\mathrm{Moonlight}} = \sum_{h=1}^{\infty} \alpha^{h-1} \kappa_{p,q}(\beta_1^h). \tag{32}$$

The series is finite because (21) gives $c_h \leq \beta_1^h$. The negative sign in (28) means that, in stationarity, the current update is negatively correlated with the current weight, and the size of this negative correlation is controlled by $C_\alpha$. Substituting (31) into (28) recovers the familiar AdamW expression

$$\gamma = -\eta C_\alpha^{\mathrm{Adam}} = -\frac{\eta\beta_1}{1 - \alpha\beta_1}. \tag{33}$$

### B.3 Equilibrium weight norm and angular step estimate

Let $S_t := \mathbb{E}\,\|W_t\|_{\mathrm{F}}^2$ and let $U^2$ be the update second moment in (7). Squaring (2), taking expectations, and substituting $\mathbb{E}\,\langle W_t,\, u_t\rangle = \gamma U^2$ gives

$$S_{t+1} = \alpha^2 S_t + \eta^2 U^2 - 2\alpha\eta\gamma U^2. \tag{34}$$

At stationarity, $S_{t+1} = S_t = R_\star^2$, and (28) gives (9). For AdamW, (31) reduces this to the closed form

$$R_\star^{\mathrm{Adam}} = \eta U \sqrt{\frac{1+\alpha\beta_1}{(1-\alpha^2)(1-\alpha\beta_1)}}. \tag{35}$$

For SVD Muon and Moonlight, the correct formula is instead (9) with $C_\alpha$ from (32).

**Corollary B.2** (Cosine proxy and angular step estimate at equilibrium)**.** *Under assumptions 4.1 and 4.2 and constant hyperparameter training, the cosine proxy between the weight and update* $\cos_t := \frac{\mathbb{E}\langle W_t,\, u_t\rangle}{R_\star U}$ *is identically equal to a constant value* $\cos_\star$ *for all t, with*

$$\cos_\star = -C_\alpha \sqrt{\frac{1-\alpha^2}{1+2\alpha C_\alpha}}. \tag{36}$$

*The corresponding equilibrium angular step estimate* $\eta_{\mathrm{eq}}^{\mathrm{ang}}$ *is given by (9), with the AdamW specialization (10).*

*Proof.* The cosine formula follows from $\mathbb{E}\,\langle W_t,\, u_t\rangle = \gamma U^2$, $\gamma = -\eta C_\alpha$, and (9):

$$\cos_\star = \frac{\gamma U}{R_\star} = -C_\alpha \sqrt{\frac{1-\alpha^2}{1+2\alpha C_\alpha}}. \tag{37}$$

For the equilibrium angular step estimate, set $k_\star := U/R_\star$ and define the stationary directional cosine proxy $\chi_\star := \mathbb{E}\,\langle W_{t+1},\, W_t\rangle\,/R_\star^2$. Using (2) and the stationary second moments gives

$$\chi_\star = \frac{\alpha - \eta\gamma k_\star^2}{\sqrt{\alpha^2 - 2\alpha\eta\gamma k_\star^2 + \eta^2 k_\star^2}}. \tag{38}$$

At equilibrium, the denominator equals 1 by the stationary radius equation. Using $k_\star^2 = (1-\alpha^2)/(\eta^2(1+2\alpha C_\alpha))$ and $\gamma = -\eta C_\alpha$ gives

$$(\eta_{\mathrm{eq}}^{\mathrm{ang}})^2 = 2 - 2\chi_\star = 2 - 2\left(\alpha + \frac{C_\alpha(1-\alpha^2)}{1+2\alpha C_\alpha}\right) = \frac{2(1-\alpha)\big(1-(1-\alpha)C_\alpha\big)}{1+2\alpha C_\alpha}. \tag{39}$$

Substituting $C_\alpha = \beta_1/(1-\alpha\beta_1)$ gives (10). □

### B.4 Proof of Theorem 4.4

**Theorem 4.4** (Weight decay sets equilibrium radius and angular step estimate)**.** *Under assumptions 4.1 and 4.2 and constant hyperparameter training with* $\alpha = 1-\eta\lambda$*, the second moment* $\mathbb{E}\,\|W_t\|_{\mathrm{F}}^2$ *of the weight in the decoupled weight decay update (2) converges to the squared equilibrium radius* $R_\star^2$*, with the equilibrium angular step estimate* $\eta_{\mathrm{eq}}^{\mathrm{ang}} := \sqrt{\mathbb{E}\,\|W_{t+1}-W_t\|_{\mathrm{F}}^2}/R_\star$ *given by*

$$R_\star = \eta U \sqrt{\frac{1+2\alpha C_\alpha}{1-\alpha^2}}, \qquad (\eta_{\mathrm{eq}}^{\mathrm{ang}})^2 = \frac{2(1-\alpha)\big(1-(1-\alpha)C_\alpha\big)}{1+2\alpha C_\alpha}. \tag{9}$$

*For idealized AdamW,* $C_\alpha = \beta_1/(1-\alpha\beta_1)$ *and*

$$\eta_{\mathrm{eq}}^{\mathrm{ang}} = \sqrt{\frac{2(1-\alpha)(1-\beta_1)}{1+\alpha\beta_1}} = \sqrt{\frac{2\eta\lambda(1-\beta_1)}{1+(1-\eta\lambda)\beta_1}}. \tag{10}$$

*Proof.* By lemma B.1, $\mathbb{E}\,\langle W_t,\, u_t\rangle = \gamma U^2$ with $\gamma = -\eta C_\alpha$. Since $0 < \alpha < 1$, the radial recursion (34) is a contraction whose fixed point is (9), and corollary B.2 gives the equilibrium angular step estimate in (9) with the AdamW specialization (10). □

### B.5 Proof of Lemma 4.5

**Lemma 4.5** (Inverse gradient scaling for scale-invariant losses)**.** *If $L : \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}} \to \mathbb{R}$ is differentiable and scale-invariant, $L(cW) = L(W)$ for all $c > 0$, then $\nabla_W L(cW) = \frac{1}{c} \nabla_W L(W)$ for all $c > 0$.*

*Proof.* Scale invariance gives $L(cW + c\epsilon) = L(W + \epsilon)$ for every matrix $\epsilon$. Differentiating both sides with respect to $\epsilon$ at $\epsilon = 0$ gives

$$\langle \nabla_W L(cW),\, c\epsilon \rangle = \langle \nabla_W L(W),\, \epsilon \rangle \qquad \forall \epsilon, \tag{40}$$

which forces $c \nabla_W L(cW) = \nabla_W L(W)$. □

### B.6 Interpretation

The main message of the theory is that decoupled weight decay should be understood as an indirect controller of angular optimization speed rather than merely as a regularizer. Lemma 4.3 shows that for common optimizers the update norm $U$ and the correlation between updates at different steps depend only on the optimizer and its hyperparameters. Theorem 4.4 gives the stationary root mean square radius $R_\star$ and equilibrium angular step estimate $\eta_{\text{eq}}^{\text{ang}}$, both determined by the optimizer and its hyperparameters. Under group scale invariance, the group angular step size measures movement after removing the redundant common scale. When relative radii are stable, the matrix effective learning rates approximate the group angular step size under the additional assumptions in section 4.2. Lemma 4.5 shows that, for scale-invariant losses, a larger equilibrium radius implies a smaller gradient norm, $\|\nabla_W L(W_t)\|_{\text{F}} \propto 1/R_t$, when the direction remains unchanged. Thus, weight decay has two coupled effects: it fixes the radial scale $R_\star$, and that radial scale sets the effective learning rate $\eta_t \|u_t\|_{\text{F}} / R_\star$. This is the mechanism that Hyperball makes explicit: instead of letting weight decay indirectly determine both the matrix norm and the effective learning rate, Hyperball fixes the norm and prescribes its effective learning rate through $\eta_t$.

## C Additional Figures

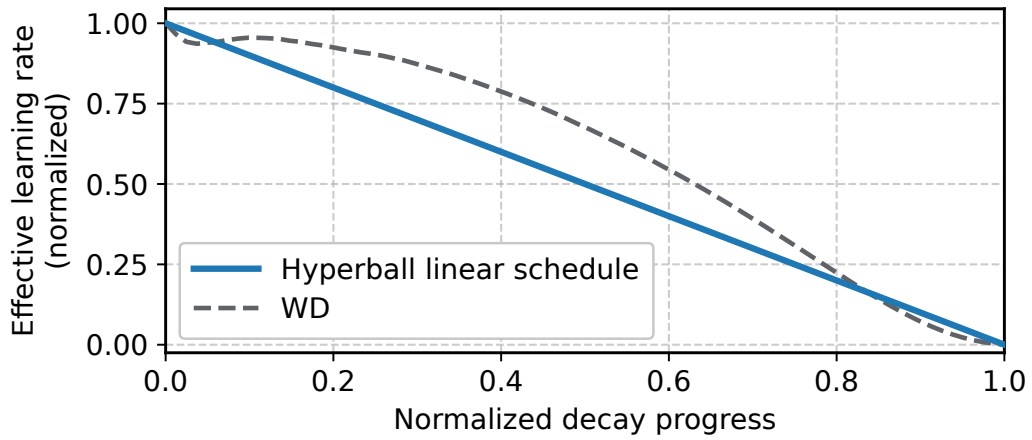


Figure 10: **Effective learning rate schedules of Hyperball and MuonW.** Prescribed $\eta_t$ versus measured $\eta_t \|u_t\|_{\text{F}} / \|W_t\|_{\text{F}}$, each normalized by its peak.

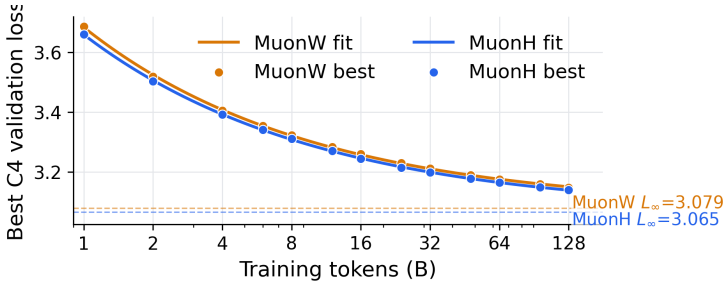


Figure 11: Overtrained 130M data scaling sweep (section 3.4). Points show the best C4 validation loss over learning rate sweeps at each token budget. Solid curves fit $L(N) = L_\infty + AN^{-\alpha}$. Dashed lines mark the fitted asymptotes.

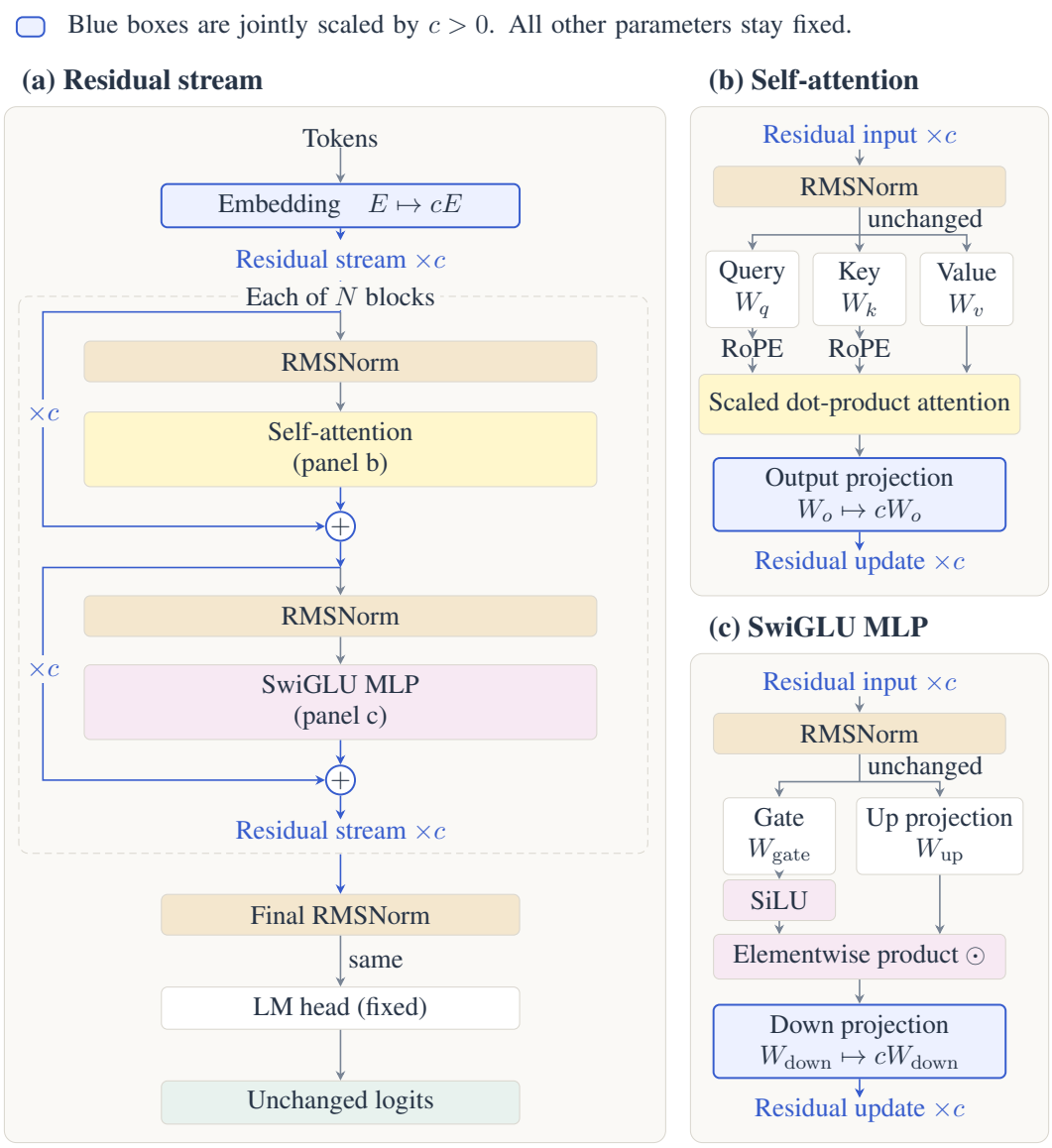


Figure 12: **Group scale invariance in a prenorm Transformer.** Jointly scaling the blue matrices by $c > 0$ scales the residual stream, and the final normalization leaves the output logits unchanged. The model has bias-free layers, an untied output head, no unscaled additive residual inputs, and zero normalization epsilon.

## D ADDITIONAL RELATED WORK

**Weight decay in normalized networks.** In normalized networks, weight decay changes optimization dynamics or effective learning rates rather than acting as a classical capacity penalty (van Laarhoven, 2017; Zhang et al., 2019; Hoffer et al., 2018; D'Angelo et al., 2023): with BatchNorm (Ioffe & Szegedy, 2015) or LayerNorm (Ba et al., 2016), it sets an equilibrium weight norm and, jointly with the learning rate, an angular step size (Li et al., 2020; Roburin et al., 2020; Kosson et al., 2024a), and Yang et al. (2023) formalize the role of the relative update size in feature learning at scale via a spectral condition. Our analysis in section 4 is closest to the rotational equilibrium framework of Kosson et al. (2024a); Hyperball prescribes the effective learning rate directly.

**Norm constraints on weights and updates.** Decoupling weight magnitude from direction has a long history, including Weight Normalization (Salimans & Kingma, 2016), Weight Standardization and BiT (Qiao et al., 2019; Kolesnikov et al., 2020), Convolutional Normalization (Liu et al., 2021), Decoupled Networks (Liu et al., 2018), Artificial Kuramoto Oscillatory Neurons (Miyato et al., 2025), the column weight normalization of EDM2 (Karras et al., 2023), and Spectral Normalization (Miyato et al., 2018). On the update side, AdamP and SGDP (Heo et al., 2021) project updates onto the tangent space of the weight direction, Lion (Chen et al., 2023) fixes the per entry update magnitude via the sign function, and LionAR (Kosson et al., 2024b) normalizes early update sizes using angular criteria. Hyperball differs in being an *optimizer wrapper* that simultaneously fixes the matrix Frobenius norm and the update Frobenius norm, exposing the effective learning rate as a prescribed quantity.