CONSTRUCTING ADAPTIVE DIRECTIONS FOR MATRIX ORTHOGONALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Coordinate-wise adaptation and matrix orthogonalization exploit complementary information—gradient history and matrix structure—but their interaction depends critically on the matrix presented to orthogonalization. We propose Relativistic Adaptive Gradient Descent with Orthogonalization (RADO), which constructs this matrix in three stages: dynamic damping forms an adaptive direction, signed-log compresses its coordinate magnitudes while retaining their ordering, and a finite Newton–Schulz transformation acts last. This graded compression preserves dependence on adaptive scaling that is eliminated by a hard sign map. Across independently tuned coordinate-adaptive and orthogonalized-momentum baselines, RADO achieves the highest mean peak ViT validation accuracy and the lowest mean final perplexity on both GPT-2 tasks. On WikiText-103, RADO reduces mean perplexity by approximately 7.45% relative to Muon and 6.57% relative to AdamW, with a 32.60% reduction in interpolated steps to Adam's final evaluation-loss target. In LLaVA fine-tuning, RADO attains the highest mean score on six of seven reported benchmarks. Component ablations and comparisons with alternative update constructions favor RADO in the tested setting, while resource measurements characterize its computational costs.

1 INTRODUCTION

Neural-network training devotes substantial computation to repeated gradient evaluations. The optimizer determines how those gradients change the model, and hence the loss and predictive performance attainable within a fixed budget. Improving the update rule can therefore benefit training without changing the architecture or data. Useful information appears at two levels: the uneven and changing gradient scales across parameters, and the matrix structure of weights that jointly implement a layer. An optimizer can draw on both when deciding how to update the model.

Two approaches address these aspects at different levels. Coordinate-wise adaptive optimizers use gradient history to adjust relative step sizes, as in Adam and AdamW (Kingma & Ba, 2015; Loshchilov & Hutter, 2019). Matrix optimizers act on relationships across entries: Muon, for example, approximately orthogonalizes momentum to reshape its singular-value distribution (Jordan et al., 2024). Combining these approaches could let gradient history guide an update while controlling its matrix geometry, and recent adaptive extensions explore this possibility (Li et al., 2026; Si et al., 2025). The central question is: *how should adaptive information enter an orthogonalized matrix update?* This interface is consequential because orthogonalization is not independent of how its input is constructed. Coordinate-wise adaptation can alter the matrix direction presented to the spectral transformation, whereas a positive scalar applied afterward changes only that tensor's gradient-update length. The problem is therefore not simply whether to combine adaptation and orthogonalization, but how much adaptive magnitude information should be retained in the matrix input.

Retaining adaptive magnitudes lets gradient history influence that input. Large contrasts can give a few entries a strong influence over the singular directions, which singular-value normalization does not undo. Keeping only signs removes such contrasts but also discards the relative scaling supplied by adaptation. Compression offers an intermediate choice: preserve adaptive magnitude distinctions while tempering the largest contrasts before constructing the matrix update.

We propose Relativistic Adaptive Gradient Descent with Orthogonalization (RADO), an optimizer that constructs adaptive directions for matrix orthogonalization. RADO first uses gradient moments

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.

and dynamic damping to set coordinate-wise update scales. Signed-log then reduces magnitude contrasts while retaining signs and magnitude ordering. Finally, finite Newton–Schulz iterations transform the compressed adaptive direction into a matrix update. Adaptation and compression determine the matrix submitted to orthogonalization, while scalar factors set the final step size. This ordering makes adaptive direction construction an explicit part of matrix optimization.

Our contributions are:

1. We develop RADO around a design principle: adaptive statistics construct the matrix input before orthogonalization. Dynamic damping and graded signed-log compression shape this input, a finite Newton–Schulz transformation controls its matrix geometry, and scalar factors set the final step size. Parameters outside the matrix branch use the adaptive direction directly.
2. We characterize the intermediate regime created by graded compression. Signed-log reduces magnitude contrasts while preserving signs, ordering, and dependence on adaptive scaling. Operation-placement analysis separates direction construction from final scalar step-size control, explaining why rescaling the input before compression differs from rescaling the completed update.
3. We evaluate RADO through fixed-budget predictive quality, component ablations and alternative update constructions, target-loss progress, and multimodal transfer. RADO attains the best mean primary vision and language scores among the evaluated baselines; resource measurements and task-specific time estimates characterize the computational trade-offs associated with these gains.

2 Preliminaries and Related Work

2.1 Adaptive Scaling and Matrix Orthogonalization

Let $W \in \mathbb{R}^{m\times n}$ be a parameter matrix and G_t its gradient at step t. A momentum estimate M_t aggregates gradient history. Coordinate-wise adaptation forms a direction $M_t \oslash D_t$, where the positive tensor D_t sets a separate scale for each entry and $\oslash$ denotes element-wise division. AdaGrad uses accumulated squared gradients to adapt coordinate scales (Duchi et al., 2011). Adam uses exponential moving averages and a stabilizer; AdamW decouples weight decay from the adaptive update (Kingma & Ba, 2015; Loshchilov & Hutter, 2019). This coordinate scaling changes the relative entries of the update matrix.

Orthogonalization instead acts on its singular-value structure. For a matrix $A = L\Sigma R^\top$ of rank $\min(m, n)$, written using a thin singular-value decomposition, the polar factor (Higham, 2008, Chapter 8) is

$$\mathrm{Orth}(A) = LR^\top. \tag{1}$$

It replaces singular values by one while retaining the singular-vector factors. A positive scalar rescaling leaves this factor unchanged, whereas coordinate-wise rescaling can change it. Thus an adaptive denominator can influence the orthogonalized direction through the matrix's relative entries, even though the polar map removes its singular-value magnitudes. Muon uses finite iterations to approximate this operation (Jordan et al., 2024). RADO uses a five-step Newton–Schulz transformation, NS5, to reshape the singular-value distribution; its finite polynomial and arithmetic conventions are specified in Section 3.3.

2.2 Design Space of Adaptive Matrix Updates

Adaptive and structured updates. Adam and AdamW use gradient moments for coordinate-wise adaptation; RAD develops a damped normalization from relativistic dynamics (Kingma & Ba, 2015; Loshchilov & Hutter, 2019; Lyu et al., 2025). K-FAC and Shampoo use structured preconditioning to account for interactions across coordinates (Martens & Grosse, 2015; Gupta et al., 2018), while Muon applies a finite approximation to polar orthogonalization to matrix momentum (Jordan et al., 2024). RADO couples coordinate-wise adaptation to a matrix transformation by constructing the adaptive direction that is orthogonalized.

Where adaptation enters. NorMuon accumulates row-wise second moments of the orthogonalized update, then normalizes and rescales it; its Front variant uses gradient-based row statistics before orthogonalization (Li et al., 2026). AdaMuon applies sign to momentum before orthogonalization and uses element-wise statistics of the transformed output for subsequent adaptation (Si et al., 2025). NAMO and NAMO-D use gradient-norm statistics for scalar or column-wise scaling of orthogonalized momentum (Zhang et al., 2026). RADO uses element-wise gradient-history statistics to shape the matrix input and applies only scalar scaling after the matrix transformation.

Controlling the matrix input. ROOT clips momentum before applying its adaptive Newton map (He et al., 2025), while AdaMuon's sign input removes magnitude distinctions. RADO uses signed-log to compress an adaptive direction: distinct magnitudes remain distinct and ordered, including beyond the saturation threshold of a hard clip. This gives RADO graded control over the adaptive magnitudes presented to NS5.

3 Constructing Adaptive Directions for Orthogonalization

RADO constructs an adaptive direction U_t from gradient history, compresses its coordinate magnitudes, and then applies finite orthogonalization (Figure 1). These stages separate coordinate-wise adaptation from the matrix transformation. Parameters excluded from the matrix branch use U_t directly.

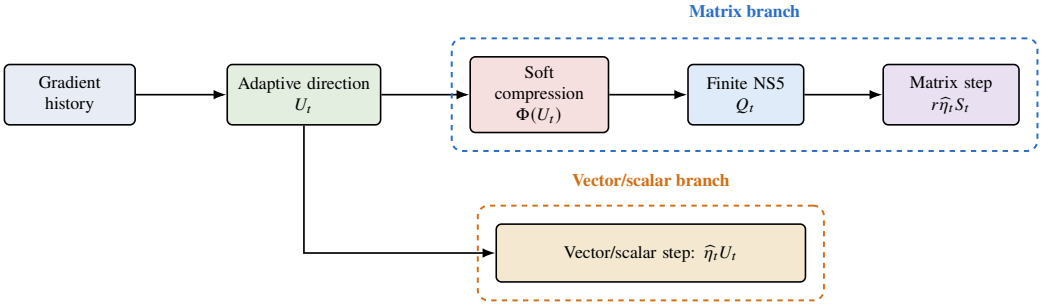


Figure 1: RADO's update construction. Gradient history and dynamic damping determine U_t. Signed-log moderates its coordinate contrasts before the eligible matrix branch applies five-step Newton–Schulz orthogonalization (NS5). Vectors, scalars, and task-specific excluded matrices use U_t directly.

3.1 Adaptive Direction with Dynamic Damping

For a parameter tensor W, let G_t be the gradient evaluated at the parameters before step t. RADO stores two tensors of the same shape, initialized as $M_0 = V_0 = 0$. With $0 \le \beta_1, \beta_2 < 1$, the first and second moments are

$$M_t = \beta_1 M_{t-1} + (1-\beta_1) G_t, \qquad V_t = \beta_2 V_{t-1} + (1-\beta_2)(G_t \odot G_t). \tag{2}$$

Small second moments increase the amplification produced by an adaptive denominator. RADO controls this amplification with a coordinate-wise damping factor computed from the previous state:

$$\delta_t = \sqrt{1 + \frac{s_\delta \,\mathrm{mean}(V_{t-1})(1-t/T)^2}{V_{t-1} + \varepsilon_\delta}}, \tag{3}$$

where $s_\delta \ge 0$ sets the damping strength, $\varepsilon_\delta > 0$ stabilizes the ratio, T is the prescribed positive integer training horizon, and the mean is over the parameter tensor. Using the damped normalization form motivated by RAD (Lyu et al., 2025), RADO constructs the adaptive direction from the *new* second moment:

$$U_t = \frac{M_t}{\sqrt{V_t \odot \delta_t^2 + \zeta}}, \qquad c_t = \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t}, \qquad \widehat{\eta}_t = \eta c_t. \tag{4}$$

Here $\odot$ and the fraction denote element-wise operations, $\zeta > 0$ is fixed within a parameter group, and η is that group's base learning rate at the current step. The bias-correction factor c_t scales the final step, outside the nonlinear transformations below.

Equation (3) uses each coordinate's past gradient energy relative to its tensor to determine the damping. At a fixed mean and time, a smaller previous second moment receives no less damping, moderating its amplification in U_t. The horizon multiplier controls the schedule, while the evolving moments determine its realized strength. The stabilizer ζ supplies a fixed denominator floor, while δ_t adjusts its gradient-dependent part. Appendix A.2 gives the schedule's endpoint behavior and the normalization's origin.

3.2 Graded Compression of Adaptive Magnitudes

The magnitudes in U_t encode coordinate-wise scaling from gradient history. If passed directly to NS5, a highly uneven U_t can give a few large entries a strong influence over the input's singular directions. Changing singular values alone preserves the corresponding singular-vector factors in exact arithmetic; a smaller outer scalar only shortens that tensor's gradient update. RADO compresses magnitudes before NS5 to temper these contrasts while retaining their ordering.

Sign-based optimization discards coordinate magnitudes, as in signSGD (Bernstein et al., 2018). Applying a sign map here would also remove the adaptive scaling: for any positive denominator D_t,

$$\operatorname{sign}(M_t \oslash D_t) = \operatorname{sign}(M_t).$$

It therefore removes the denominator's influence on the matrix input at a fixed momentum state. To compress magnitudes without collapsing them to a single nonzero level, RADO applies signed-log to tensors with at least two dimensions:

$$\phi(x) = \operatorname{sign}(x)\log(1+|x|), \qquad P_t = \Phi(U_t), \tag{5}$$

where Φ applies ϕ to each entry. The following properties characterize this choice.

Proposition 1 (Coordinate compression)**.** *With $\phi(0) = 0$, signed-log is odd, continuously differentiable, and strictly increasing, with $\phi'(x) = 1/(1+|x|)$. It preserves signs and ordering, satisfies $|\phi(x)| \le |x|$, and is 1-Lipschitz. Its element-wise extension obeys $\|\Phi(A) - \Phi(B)\|_F \le \|A - B\|_F$ for equally shaped matrices. For $a > b > 0$,*

$$1 < \frac{\log(1+a)}{\log(1+b)} \le \frac{a}{b}. \tag{6}$$

The proof is in Appendix A.3. Since its derivative decreases with input magnitude, signed-log is approximately linear near zero and increasingly compressive for larger entries. Equation (6) shows that this compression reduces relative contrasts, not just the overall scale of the matrix.

Both U_t and $\Phi(U_t)$ generally retain the adaptive denominator's influence on the matrix input. The additional role of signed-log is to moderate the contrasts through which that influence acts. This is the reason for placing compression between adaptive scaling and orthogonalization: it controls the matrix being transformed while preserving graded coordinate information.

The input contrast also has an interpretable dependence on overall magnitude. For fixed $a > b > 0$ and a common rescaling $\kappa > 0$, define

$$R_{a,b}(\kappa) = \frac{\log(1+\kappa a)}{\log(1+\kappa b)}, \qquad \lim_{\kappa\to 0^+} R_{a,b}(\kappa) = \frac{a}{b}, \qquad \lim_{\kappa\to\infty} R_{a,b}(\kappa) = 1. \tag{7}$$

Thus small input magnitudes retain nearly uncompressed ratios, while at large common scales the ratio approaches one. At every finite scale, distinct positive magnitudes remain distinct. Signed-log therefore provides a graded regime between passing adaptive magnitudes through unchanged and removing them altogether. Appendix A.3 derives these limits.

3.3 Finite Matrix Transformation

RADO applies five Newton–Schulz iterations to each compressed matrix P_t. The procedure first converts the input to BF16 and transposes its last two axes when rows exceed columns. Each resulting matrix is divided by its Frobenius norm plus 10^{-7} to obtain X_0. The iterations are

$$A_j = X_j X_j^\top, \qquad X_{j+1} = 3.4445 X_j + (-4.7750 A_j + 2.0315 A_j^2) X_j, \quad j = 0, \dots, 4, \tag{8}$$

using the polynomial coefficients also employed in Muon (Jordan et al., 2024). After restoring the original orientation, RADO applies aspect-ratio scaling:

$$Q_t = \mathrm{NS5}(P_t), \qquad S_t = \sqrt{\max(1, m/n)}\, Q_t, \tag{9}$$

where m, n are the matrix-view dimensions before the internal transpose. Equation (8) defines the finite spectral transformation used in the update. In exact arithmetic, the polynomial acts along the input's singular-vector factors; Appendix A.4 characterizes its singular-value action and its relationship to a polar reference.

3.4 Operation Placement and the Final Update

Coordinate adaptation, magnitude compression, and matrix transformation control different aspects of the update. As Section 2.1 shows, coordinate scaling before polar orthogonalization can change the selected matrix direction; applied afterward, it can change the transformed spectrum. RADO places its coordinate operations before NS5, so they construct the matrix whose spectral structure is transformed.

Scalar scaling also depends on placement. The input contrasts in Eq. (7) vary with a common rescaling, so $\Phi(c_t U_t)$ need not be collinear with $\Phi(U_t)$. Keeping bias correction outside Φ lets c_t adjust the final step size without changing the compressed input. The matrix ratio r likewise scales the completed direction. At a fixed optimizer state, positive outer scalars preserve each tensor's gradient-update direction; weight decay and relative scaling across parameter groups affect the complete parameter displacement.

This ordering distinguishes two design decisions. Replacing signed-log with identity or sign changes which magnitude distinctions reach NS5; moving coordinate adaptation across NS5 changes whether those statistics act on its input or its output. These changes address distinct design questions; isolating placement requires holding the statistic source and final scalar scaling fixed.

Let $g(W) \in \{0, 1\}$ indicate the parameter group's matrix-branch eligibility. After decoupled weight decay $W \leftarrow (1 - \eta\lambda)W$, the gradient update is

$$W \leftarrow \begin{cases} W - r\widehat{\eta}_t S_t, & g(W) = 1 \text{ and } \mathrm{ndim}(W) \geq 2, \\ W - \widehat{\eta}_t U_t, & \text{otherwise.} \end{cases} \tag{10}$$

The ratio r sets the matrix branch's learning-rate scale relative to the standard branch, which uses U_t directly. Within the matrix branch, four-dimensional convolution kernels are flattened into matrices; other higher-dimensional tensors use their last two axes as matrix axes. Appendix A specifies shape restoration and parameter-group conventions.

Algorithm 1 RADO

Input: W, group eligibility $g(W)$, loss $\mathcal{L}$, base learning rate η, matrix ratio r, moments (β_1, β_2), decay λ, stabilizer ζ, damping parameters $(s_\delta, \varepsilon_\delta)$, horizon T
Initialize $M_0 \leftarrow 0$, $V_0 \leftarrow 0$
for $t = 1, 2, \ldots$ **do**
  $G_t \leftarrow \nabla_W \mathcal{L}(W)$; $W \leftarrow (1 - \eta\lambda)W$
  Compute δ_t from V_{t-1} using Eq. (3)
  Update M_t, V_t using Eq. (2)
  $U_t \leftarrow M_t \oslash \sqrt{V_t \odot \delta_t^2 + \zeta}$; $\widehat{\eta}_t \leftarrow \eta\sqrt{1 - \beta_2^t}/(1 - \beta_1^t)$
  if $g(W) = 1$ and $\mathrm{ndim}(W) \geq 2$ **then**
    Flatten U_t if 4-D; record matrix-view dimensions m, n
    $Q_t \leftarrow \mathrm{NS5}(\Phi(U_t))$
    $S_t \leftarrow \sqrt{\max(1, m/n)}\, Q_t$; restore S_t to the shape of W
    $W \leftarrow W - r\widehat{\eta}_t S_t$
  else
    $W \leftarrow W - \widehat{\eta}_t U_t$
  end if
end for

Algorithm 1 summarizes the update. Appendix B provides a conditional descent bound under bounded update energy and expected alignment with the objective gradient.

4 EXPERIMENTAL EVIDENCE

We evaluate fixed-budget predictive quality and multimodal transfer, examine component and operation-placement variants, and quantify optimization progress and resource costs. The tasks cover GPT-2 training with WikiText-103 and Alpaca (Radford et al., 2019; Merity et al., 2017; Taori et al., 2023), ViT on CIFAR100 (Dosovitskiy et al., 2021; Krizhevsky, 2009), and LLaVA-v1.5-7B with LoRA fine-tuning (Liu et al., 2024a; Hu et al., 2022). The first three tasks compare six optimizers over five seeds; LLaVA compares AdamW, Muon, and RADO over three seeds.

Baselines and reporting. Adam and AdamW provide coordinate-adaptive baselines, RAD provides damped coordinate-wise adaptation, and Muon provides orthogonalized momentum. SGD supplies a non-adaptive reference. We use optimizer-specific configurations selected through separate tuning runs; Appendix C reports the search spaces and task-specific selection rules. Tables report arithmetic means and sample standard deviations, with perplexity computed per run before averaging.

4.1 PREDICTIVE PERFORMANCE AT FIXED BUDGETS

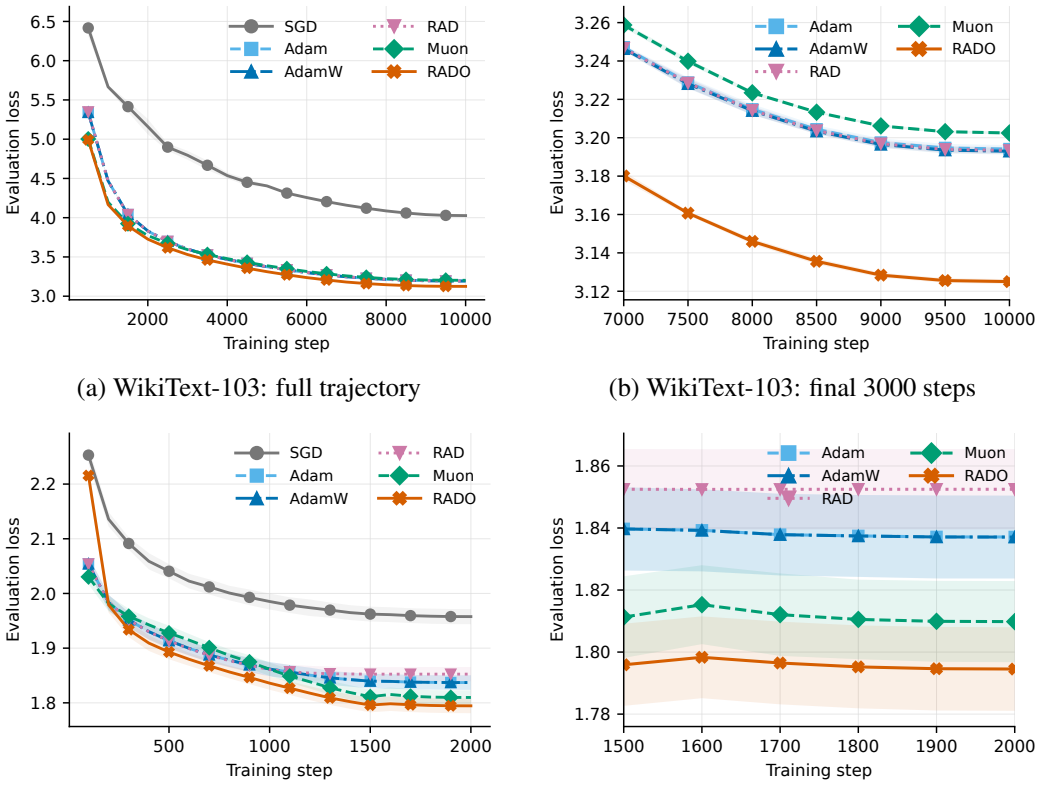
Table 1: Evaluation outcomes for the selected configurations (five seeds, mean $\pm$ sample standard deviation). ViT reports each seed's maximum validation accuracy; Alpaca and WikiText-103 report final perplexity at steps 2000 and 10,000. Bold indicates the best mean within each column. Full metrics and selection details appear in Appendix D.

Optimizer	ViT accuracy (%) $\uparrow$	Alpaca PPL $\downarrow$	WikiText-103 PPL $\downarrow$
SGD	59.53 ± 0.74	7.083 ± 0.098	56.085 ± 0.458
Adam	65.01 ± 0.54	6.279 ± 0.084	24.388 ± 0.051
AdamW	64.44 ± 0.48	6.279 ± 0.084	24.361 ± 0.042
RAD	63.49 ± 0.52	6.376 ± 0.083	24.371 ± 0.043
Muon	68.04 ± 0.43	6.110 ± 0.080	24.593 ± 0.018
RADO	$\mathbf{68.97 \pm 0.47}$	$\mathbf{6.017 \pm 0.081}$	$\mathbf{22.760 \pm 0.030}$

Language modeling and instruction tuning. WikiText-103 uses a common budget of 10,000 optimizer steps for a GPT-2 configuration with 12 layers, 12 attention heads, and hidden size 768. RADO reaches mean final perplexity 22.760, compared with 24.371 for RAD, 24.388 for Adam, and 24.593 for Muon (Table 1). The reduction in mean perplexity is approximately 7.45% relative to Muon and 6.57% relative to AdamW. On Alpaca at step 2000, RADO again has the lowest mean final perplexity, 6.017 ± 0.081, compared with 6.110 ± 0.080 for Muon.

Figure 2 pairs full evaluation histories with late-training views. WikiText-103 is evaluated every 500 steps; both language comparisons use the common final checkpoint. These trajectories also support the target-loss comparison below.

Vision performance and training fit. On ViT–CIFAR100, RADO reaches the highest mean peak validation accuracy, $68.97 \pm 0.47\%$, compared with $68.04 \pm 0.43\%$ for Muon. Each seed is selected by its maximum logged accuracy over 100 epochs, with ties resolved by the earliest epoch; RADO's mean validation loss at those records is 1.8714. Training fit and evaluation quality need not rank methods alike: on Alpaca, RADO has lower final evaluation perplexity but higher final logged training loss than Muon. Appendix D gives the full tables and training curves.



(a) WikiText-103: full trajectory

(b) WikiText-103: final 3000 steps

(c) Alpaca: full trajectory

(d) Alpaca: final 500 steps

Figure 2: Language-task evaluation trajectories. Lines are five-seed means; bands are ± 1 sample standard deviation. Full trajectories show progress across the training budget; late views resolve the final comparisons and omit SGD.

4.2 Transfer to Multimodal Fine-Tuning

Using the official LLaVA training framework (Liu et al., 2024a), we fine-tune LLaVA-v1.5-7B with LoRA on a fixed 80,000-example subset for one epoch (5000 steps), with effective batch size 16. The subset construction is described in Appendix C. Table 2 reports seven downstream benchmarks, defined with their sources in Appendix C.4. RADO leads MME, MMBench, ScienceQA, TextVQA, GQA, and VQAv2; on VQAv2 test-dev it reaches 77.68 ± 0.11, versus 77.46 ± 0.05 for Muon and 76.94 ± 0.01 for AdamW. These gains extend the evaluation beyond vision classification and text generation to visual question answering and multimodal reasoning. AdamW leads POPE F1.

Table 2: LLaVA-v1.5-7B LoRA results (three seeds, mean $\pm$ sample standard deviation; higher is better). Bold indicates the best mean per metric. Each optimizer uses its selected LoRA update scale and weight decay (Appendix C).

Metric	AdamW	Muon	RADO
POPE F1	$\mathbf{85.39 \pm 0.98}$	85.19 ± 0.96	85.15 ± 0.72
MME	1710.84 ± 36.48	1755.30 ± 38.06	$\mathbf{1779.03 \pm 21.14}$
MMBench (local dev)	72.55 ± 0.31	73.16 ± 0.49	$\mathbf{73.49 \pm 0.43}$
ScienceQA (all)	69.74 ± 0.84	69.76 ± 0.60	$\mathbf{70.59 \pm 0.52}$
TextVQA	54.70 ± 0.05	55.30 ± 0.64	$\mathbf{55.85 \pm 0.17}$
GQA	60.91 ± 0.29	61.01 ± 0.28	$\mathbf{61.13 \pm 0.10}$
VQAv2 (test-dev)	76.94 ± 0.01	77.46 ± 0.05	$\mathbf{77.68 \pm 0.11}$

The comparison uses separately selected LoRA update scales and optimizer-specific weight decay. Downstream benchmarks also informed configuration selection (Appendix C); a separate training-loss progress analysis appears in Appendix D.4.

4.3 Ablation Studies

We compare damping, compression, and operation-placement variants using single-run trajectories from a 124M-parameter GPT-2 trained on WikiText-103 for 10,000 updates. Training settings are shared without separate tuning; the two placement variants also use a different damping strength (Appendix C.3).

Let H_t denote the element-wise preconditioner, so the adaptive direction is $U_t = M_t \odot H_t$. Each variant produces a matrix direction $Q_t^{(v)}$ before common scalar scaling; full RADO uses $Q_t = \mathrm{NS5}(\Phi(U_t))$. The first four variants examine damping and compression under a common precondition-first ordering; the remaining two evaluate alternative update constructions that jointly modify operation placement and damping strength. Table 5 gives the six paths and implementation identifiers.

Full RADO attains the lowest validation loss at every checkpoint shown in Figure 3.

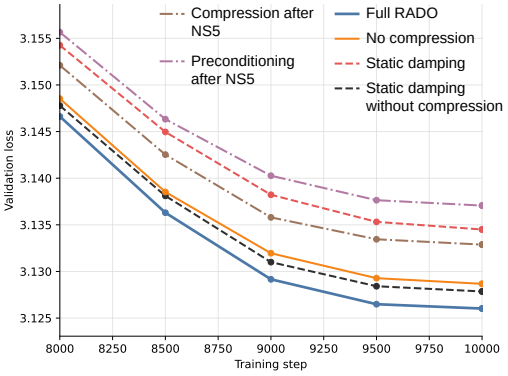


Figure 3: GPT-2/WikiText-103 ablation validation loss (seed 42); protocol in Appendix C.3.

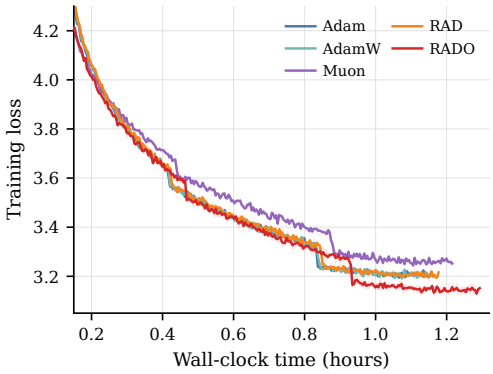


Figure 4: WikiText-103 training loss versus estimated wall-clock time: five-seed mean ± sample SD; steps are scaled by each run's measured duration (Appendix C.5).

4.4 Steps, Estimated Time-to-Target, and Resource Costs

To test whether RADO's predictive gains are accompanied by earlier optimization progress, we measure the updates needed to reach Adam's final loss on Alpaca and WikiText-103. For each task, the target is Adam's final seed-mean evaluation loss. Let τ_o be optimizer o's crossing coordinate and T_{Adam} the reference budget. The reported reduction is

$$100\,\frac{T_{\mathrm{Adam}} - \tau_o}{T_{\mathrm{Adam}}}. \tag{11}$$

Adam's row records the reference budget endpoint; the other attained entries are first crossings. For both tasks, the crossing is linearly interpolated between adjacent recorded points on the seed-mean evaluation-loss curve and rounded to the nearest integer step; the reduction is computed from that displayed integer. No additional smoothing is applied. Appendix D gives the targets and logging definitions.

Table 3: (a) Language-task target attainment and estimated time; "–" denotes an unreached target. (b) Single-GPU LLaVA resource measurements; training-step $\pm$ values are standard deviations across 300 measured steps within one run per optimizer. Reductions in panel (a) are relative to Adam; protocols and calculations appear in Appendix C.5.

(a) Language-task target attainment and estimated time

Optimizer	Alpaca: step (reduction)	WikiText: step (reduction)	Alpaca: est. min (reduction)	WikiText: est. min (reduction)
SGD	–	–	–	–
Adam	2000 (0.00%)	10000 (0.00%)	18.07 (0.00%)	69.64 (0.00%)
AdamW	1916 (4.20%)	9411 (5.89%)	17.30 (4.29%)	65.83 (5.48%)
RAD	–	9481 (5.19%)	–	67.16 (3.57%)
Muon	1213 (39.35%)	–	11.26 (37.69%)	–
RADO	**990 (50.50%)**	**6740 (32.60%)**	**9.66 (46.53%)**	**52.41 (24.74%)**

(b) Single-GPU LLaVA resource measurements (lower is better)

Optimizer	Memory cost of optimizer states (GB)	Profiled optimizer GPU time (s)	Training step time (s)
AdamW	1.363	0.011	5.333 ± 0.596
Muon	0.724	0.384	5.534 ± 0.595
RADO	1.363	0.541	5.623 ± 0.597

Table 3 reports RADO's interpolated target crossings at step 990 on Alpaca and step 6740 on WikiText-103, corresponding to reductions of 50.50% and 32.60% against Adam's reference budgets.

Resource costs and estimated time to target. In the single-GPU LLaVA probe, RADO uses 1.363 GB of optimizer-state memory, matching AdamW. Its mean training-step time is 5.44% longer than AdamW's and 1.61% longer than Muon's. For the language tasks, we estimate target time as the displayed crossing step times the task-specific mean seconds per step over five runs, divided by 60. The estimated time reductions are 46.53% on Alpaca and 24.74% on WikiText-103. Appendix C.5 gives the timing and profiler definitions.

Figure 4 shows training loss against estimated wall-clock time, obtained by distributing each run's measured duration uniformly across its steps. RADO has the lowest training loss among the plotted optimizers late in this interval.

5 Conclusion

RADO connects gradient-history adaptation with matrix orthogonalization through the choice of matrix input. Dynamic damping forms the adaptive direction, signed-log moderates its coordinate contrasts, and finite Newton–Schulz iterations produce the matrix update. The analysis explains how graded compression retains input distinctions that sign removes. RADO improves mean peak ViT accuracy and final GPT-2 perplexity in the evaluated comparisons, reaches the language-task loss targets earlier, and leads six of seven reported benchmarks in LLaVA fine-tuning. The tested component and update variants favor the chosen construction in the single-seed setting, while resource measurements and time-to-target estimates characterize its computational trade-offs.

Limitations and Outlook. The results point to a broader design space for constructing adaptive directions across architectures and training scales. Natural extensions include co-designing magnitude compression and matrix transformations for different parameter structures, evaluating the resulting updates at larger pretraining scales, and developing tighter analyses of their geometry and convergence. Accelerator-aware implementations may further translate these optimization gains into improved system-level efficiency.

Reproducibility Statement

Algorithm 1 and Appendix A specify the optimizer update, parameter routing, tensor transformations, and numerical conventions. Appendix B provides the theoretical assumptions and proofs, while Appendices C and D report experimental configurations, hyperparameter selection, aggregation rules, full results, and additional learning curves.

AI Use Statement

Generative AI tools were used for technical and editorial assistance, including code development and debugging, data aggregation and plotting, literature verification, and language polishing. All scientific decisions, methods, experiments, analyses, interpretations, and conclusions were determined and verified by the authors.

REFERENCES

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pp. 560–569. PMLR, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, Yunsheng Wu, Rongrong Ji, Caifeng Shan, and Ran He. MME: A comprehensive evaluation benchmark for multimodal large language models. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/d79a27cf2772fe00be7f341efc0eb517-Abstract-Datasets_and_Benchmarks_Track.html.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6904–6913, 2017. URL https://openaccess.thecvf.com/content_cvpr_2017/html/Goyal_Making_the_v_CVPR_2017_paper.html.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pp. 1842–1850. PMLR, 2018.

Wei He, Kai Han, Hang Zhou, Hanting Chen, Zhicheng Liu, Xinghao Chen, and Yunhe Wang. ROOT: Robust orthogonalized optimizer for neural network training, 2025. URL https://arxiv.org/abs/2511.20626v1.

Nicholas J. Higham. *Functions of Matrices: Theory and Computation*. SIAM, 2008.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2106.09685.

Drew A. Hudson and Christopher D. Manning. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6700–6709, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Hudson_GQA_A_New_Dataset_for_Real-World_Visual_Reasoning_and_Compositional_CVPR_2019_paper.html.

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

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

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 292–305, 2023. doi: 10.18653/v1/2023.emnlp-main.20. URL https://aclanthology.org/2023.emnlp-main.20/.

Zichong Li, Liming Liu, Chen Liang, Weizhu Chen, and Tuo Zhao. NorMuon: Making Muon more efficient and scalable. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/61880.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024a. URL https://arxiv.org/abs/2310.03744.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. MMBench: Is your multi-modal model an all-around player? In *European Conference on Computer Vision*, 2024b. URL https://www.ecva.net/papers/eccv_2024/papers_ECCV/html/959_ECCV_2024_paper.php.

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

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/11332b6b6cf4485b84afadb1352d3a9a-Abstract-Conference.html.

Yao Lyu, Xiangteng Zhang, Shengbo Eben Li, Jingliang Duan, Letian Tao, Qing Xu, Lei He, and Keqiang Li. Conformal symplectic optimization for stable reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 36(6):11049–11063, 2025. doi: 10.1109/TNNLS.2024.3511670.

James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *International Conference on Machine Learning*, pp. 2408–2417. PMLR, 2015.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Chongjie Si, Debing Zhang, and Wei Shen. AdaMuon: Adaptive Muon optimizer, 2025. URL https://arxiv.org/abs/2507.11005.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards VQA models that can read. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8317–8326, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Singh_Towards_VQA_Models_That_Can_Read_CVPR_2019_paper.html.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpaca: A strong, replicable instruction-following model. Stanford CRFM Blog, 2023. URL https://crfm.stanford.edu/2023/03/13/alpaca.html.

Minxin Zhang, Yuxuan Liu, and Hayden Schaeffer. Adam improves Muon: Adaptive moment estimation with orthogonalized momentum, 2026. URL https://arxiv.org/abs/2602.17080.

A Method Specification and Operator Properties

A.1 Update Ordering, Shapes, and Scaling

The gradient is evaluated at the old parameter value. Decoupled decay $W \leftarrow (1 - \eta\lambda)W$ is then applied before either gradient-update branch. The damping factor is evaluated from V_{t-1} before the moment states are updated, whereas U_t uses the new state V_t. This ordering separates the historical scale used for damping from the current scale used in the adaptive denominator.

The matrix branch requires both group eligibility $g(W) = 1$ and at least two tensor dimensions. Within this branch, a four-dimensional convolution kernel with shape $(d_{\text{out}}, d_{\text{in}}, k_h, k_w)$ is reshaped to $d_{\text{out}} \times (d_{\text{in}} k_h k_w)$ before signed-log compression and NS5, then restored to its original shape. Other higher-dimensional tensors retain their batched last-two-dimensional structure. The aspect-ratio factor uses the original matrix-view dimensions, not the temporarily transposed dimensions inside NS5. Vectors, scalars, and explicitly excluded matrices skip these operations and use the standard RADO update. Task-specific routing appears in Appendix C.

The stabilizer ζ is fixed within each parameter group. It is distinct from the damping-ratio stabilizer ε_δ and the 10^{-7} constant in NS5 normalization. The base learning rate η sets the decay factor, while the gradient update also contains c_t and, for the matrix branch, r and the aspect-ratio factor. In the non-matrix branch, writing $\widehat{M}_t = M_t/(1 - \beta_1^t)$ and $\widehat{V}_t = V_t/(1 - \beta_2^t)$ gives

$$\frac{c_t M_t}{\sqrt{V_t \odot \delta_t^2 + \zeta}} = \frac{\widehat{M}_t}{\sqrt{\widehat{V}_t \odot \delta_t^2 + \zeta/(1 - \beta_2^t)}}. \tag{12}$$

Thus replacing the denominator by $\sqrt{\widehat{V}_t \odot \delta_t^2 + \zeta}$ would change the update. In the matrix branch, moving c_t inside Φ would also generally change the direction because signed-log is not homogeneous.

A.2 Damping Schedule and Normalization

RAD distinguishes a deterministic relativistic update from its moving-average extension (Lyu et al., 2025). The deterministic coordinate rule has the form $\Delta\theta_i = -\alpha v_i/\sqrt{\delta^2 v_i^2 + 1}$ and hence $|\Delta\theta_i| \leq \alpha/\delta$ for $\delta > 0$. Its stochastic algorithm replaces the squared momentum in the denominator with a gradient second-moment estimate, includes bias correction, and schedules a stabilizer $\zeta_k = \min\{\epsilon_k, 1 - \beta_2^{k+1}\}$ (RAD, Algorithm 3). These steps are separate from the original deterministic derivation.

RADO's normalization in Eq. (4) uses a fixed group stabilizer ζ and the coordinate-wise schedule in Eq. (3). The damping strength $s_\delta \geq 0$, stabilizer $\varepsilon_\delta > 0$, tensor mean, and horizon multiplier define that schedule. The default damping parameters are $s_\delta = 10^{-4}$ and $\varepsilon_\delta = 10^{-8}$; the schedule uses no clipping. The connection to relativistic dynamics motivates the base normalization form; the state-dependent schedule and matrix-input construction are RADO's algorithmic design choices.

For nonnegative second moments, $s_\delta \geq 0$, $\varepsilon_\delta > 0$, and $T > 0$, define $a_t = s_\delta \,\mathrm{mean}(V_{t-1})(1 - t/T)^2$. Then each coordinate satisfies

$$\delta_{t,i}^2 = 1 + \frac{a_t}{V_{t-1,i} + \varepsilon_\delta}. \tag{13}$$

With zero initialization, $a_1 = 0$ and $\delta_1 = 1$; at $t = T$, $a_T = 0$ and $\delta_T = 1$. Between these endpoints, the ratio of the tensor mean to each coordinate's past second moment can change, so the decreasing time multiplier does not imply decreasing damping. If the update is continued beyond T with the same horizon, $(1 - t/T)^2$ grows again. The stated rule evaluates this expression without a post-horizon clamp; the chosen t, T and any restart policy therefore form part of the optimizer configuration.

A.3 Signed-Log Properties and the Sign Comparison

Proof of Proposition 1. Away from zero, differentiation gives the stated derivative; at zero, both difference quotients tend to one. Positivity of the derivative gives monotonicity, and $\log(1+x) \le x$ for $x \ge 0$ gives attenuation. The derivative is at most one, so the mean-value theorem gives the scalar Lipschitz inequality; squaring and summing it over entries yields the Frobenius inequality. Finally, $h(x) = \log(1+x)/x$ decreases on $x > 0$, since its derivative has numerator $x/(1+x) - \log(1+x) < 0$. Applying this to $a > b > 0$, together with strict monotonicity of $\log(1+x)$, proves the ratio bound. □

For any positive coordinate-wise denominator D, $\operatorname{sign}(M \oslash D) = \operatorname{sign}(M)$ in exact arithmetic. Thus sign removes the denominator's influence on the input matrix at a fixed state. In general, $\Phi(M \oslash D)$ still depends on D, and $\Phi(cU)$ need not be collinear with $\Phi(U)$ even for a positive scalar c. The uncompressed adaptive direction also depends on D; signed-log additionally attenuates relative magnitudes. Unlike hard clipping, it remains unbounded. These properties characterize the matrix input; the subsequent transformation and input statistics determine their effect on the complete update.

Common-scale limits. For $a > b > 0$, the expansion $\log(1+\kappa a) = \kappa a + O(\kappa^2)$ as $\kappa \to 0^+$, and its counterpart for b, give the first limit in Eq. (7). As $\kappa \to \infty$, write $\log(1+\kappa a) = \log\kappa + \log a + o(1)$; the common leading term gives the second limit. Since $a/b > 1$, the two limits differ, so the compressed magnitude ratio generally depends on a common input rescaling. Strict monotonicity of ϕ preserves the distinction between the two magnitudes at every finite $\kappa > 0$.

A.4 Finite NS5 and Its Spectral Reference

Section 3.3 specifies the BF16 conversion, orientation, normalization, and five polynomial iterations. In exact arithmetic, an iteration acts along the singular-vector factors through $f(s) = 3.4445s - 4.7750s^3 + 2.0315s^5$; the resulting singular value is $|f(s)|$. In particular, $f(1) = 0.701$, and zero singular values remain zero. For a fixed number of iterations, the output also approaches zero as an input singular value approaches zero, whereas the full-rank polar factor assigns it one. The finite polynomial and BF16 arithmetic thus define the actual matrix transformation, including its behavior near rank deficiency.

A separate analytical reference for $P \in \mathbb{R}^{m\times n}$ is the *regularized* polar map

$$\operatorname{Orth}_\mu(P) = \begin{cases} P(P^\top P + \mu I)^{-1/2}, & m \ge n, \\ (PP^\top + \mu I)^{-1/2} P, & m < n, \end{cases} \qquad \mu > 0. \tag{14}$$

Proposition 2 (Spectrum of the regularized reference)**.** *The singular values of* $\operatorname{Orth}_\mu(P)$ *are* $\sigma_i/\sqrt{\sigma_i^2+\mu}$*, where* σ_i *are those of* P*. Hence* $\|\operatorname{Orth}_\mu(P)\|_2 \le 1$ *and* $\|\operatorname{Orth}_\mu(P)\|_F^2 \le \min\{m,n\}$.

Proof. Substituting an SVD of P into either branch of Eq. (14) gives the stated singular values, each in $[0,1)$. The norm bounds follow. □

For an input of rank $\min(m,n)$, the limit $\mu \to 0$ yields the exact polar factor, whose singular values are one. The norm bounds above characterize this regularized reference. RADO uses the finite polynomial above with BF16 arithmetic and no μ parameter, giving a distinct spectral transformation whose behavior is specified in Section 3.3.

B Conditional Descent Interpretation

Consider the objective

$$\min_{\theta \in \mathbb{R}^d} J(\theta), \qquad J(\theta) = \mathbb{E}_\xi[\ell(\theta; \xi)]. \tag{15}$$

Let N denote the number of updates analyzed, distinct from the prescribed damping horizon T. Index successive updates from a fixed initial point θ_0 by $t = 0, \ldots, N-1$. Let d_t denote the complete flattened gradient-update direction used at Algorithm 1's step $t+1$, before multiplication by a common base step size. It includes bias correction, matrix ratio, aspect-ratio scaling, and both parameter branches. If parameter groups have fixed relative learning-rate multipliers, those multipliers are also included in d_t; the assumptions below then concern that complete direction. Let $\mathcal{F}_t$ be the history before sampling the current minibatch; θ_t is measurable with respect to it.

Assumption 1 (Smoothness and lower boundedness)**.** *J is L-smooth and lower bounded by $J_\star$.*

Assumption 2 (Bounded update energy)**.** *There exists $G_d > 0$ such that $\mathbb{E}[\|d_t\|^2 \mid \mathcal{F}_t] \le G_d^2$.*

Assumption 3 (Expected descent alignment)**.** *There exist $\rho > 0$ and nonnegative, integrable, $\mathcal{F}_t$-measurable errors e_t such that $\mathbb{E}[\langle \nabla J(\theta_t), d_t \rangle \mid \mathcal{F}_t] \ge \rho \|\nabla J(\theta_t)\|^2 - e_t$.*

Theorem 1 (Conditional descent bound)**.** *Under Assumptions 1–3, with zero weight decay and constant base step size $\alpha/\sqrt{N}$ for $\alpha > 0$,*

$$\frac{1}{N} \sum_{t=0}^{N-1} \mathbb{E}\|\nabla J(\theta_t)\|^2 \le \frac{J(\theta_0) - J_\star}{\rho \alpha \sqrt{N}} + \frac{L \alpha G_d^2}{2\rho\sqrt{N}} + \frac{1}{\rho N} \sum_{t=0}^{N-1} \mathbb{E}[e_t]. \tag{16}$$

Proof. By L-smoothness and $\theta_{t+1} = \theta_t - \alpha_N d_t$, where $\alpha_N = \alpha/\sqrt{N}$,

$$J(\theta_{t+1}) \le J(\theta_t) - \alpha_N \langle \nabla J(\theta_t), d_t \rangle + \frac{L \alpha_N^2}{2} \|d_t\|^2. \tag{17}$$

Taking conditional expectation, applying Assumptions 2 and 3, and then taking total expectation gives

$$\rho \alpha_N \mathbb{E}\|\nabla J(\theta_t)\|^2 \le \mathbb{E}[J(\theta_t) - J(\theta_{t+1})] + \alpha_N \mathbb{E}[e_t] + \frac{L \alpha_N^2 G_d^2}{2}. \tag{18}$$

Summing over $t = 0, \ldots, N-1$, using $J(\theta_N) \ge J_\star$, and dividing by $\rho \alpha_N N$ yields Eq. (16). □

Rate interpretation. For fixed θ_0 and α, if ρ is bounded away from zero, G_d is uniformly bounded over the analysis lengths considered, and $N^{-1} \sum_t \mathbb{E}[e_t] = O(N^{-1/2})$, Eq. (16) gives the standard $O(N^{-1/2})$ stationarity rate. The bound provides a general descent interpretation of the complete transformed direction under Assumptions 1–3; sharper RADO-specific guarantees depend on quantitative alignment and finite-transformation approximation properties.

Direction and approximation error. If $d_t^\star$ is a chosen reference direction, then

$$|\langle \nabla J(\theta_t), d_t - d_t^\star \rangle| \le \|\nabla J(\theta_t)\| \, \|d_t - d_t^\star\|. \tag{19}$$

Equation (19) separates reference alignment from transformation error, providing a route to tighter guarantees that track compression, finite matrix transformation, and numerical approximation separately.

C Experimental Protocols and Configurations

C.1 Metrics and Aggregation

Seeds and metric selection. The ViT, Alpaca, and WikiText-103 comparisons use seeds $\{42, 12345, 23456, 34567, 45678\}$; LLaVA uses $\{42, 123, 1234\}$. Each seed contributes once. Tables report arithmetic means and sample standard deviations with denominator $n - 1$. Perplexity is computed per run before averaging, and accuracy is reported in percent. ViT selects the maximum logged validation accuracy within 100 epochs, taking the earliest epoch on ties and the losses at that record. Alpaca uses the final evaluation at step 2000. WikiText-103 uses a GPT-2 configuration with 12 layers, 12 heads, and hidden size 768, evaluated every 500 steps through step 10,000. Its table uses the final evaluation and whole-run training loss. Standard deviations describe variation across seeds.

Aggregation procedure. The reported aggregates were computed from training logs, JSON result files, and LLaVA evaluations. Means, sample standard deviations, and variances were calculated before display rounding; absent observations and missing benchmark scores were not replaced by zero. The aggregation checked the expected seed set for every metric and compared each LLaVA loss with the mean of its 5000 unique step records. Learning-curve lines show seed means, with bands representing sample standard deviations across seeds.

C.2 Task Configurations and Selection Rules

Hyperparameter selection. For ViT–CIFAR100, Alpaca, and WikiText-103, each optimizer is tuned over three learning rates and two weight-decay values (Table 4). Selection uses validation accuracy for ViT and held-out evaluation loss/perplexity for the language tasks. The selected configuration is then fixed for the repeated runs. LLaVA uses separate optimizer-specific learning-rate and LoRA-multiplier tuning, described in Appendix C.4. Component ablations are separate from these hyperparameter searches.

Table 4: Optimizer-specific hyperparameter search spaces for the three five-seed tasks. Each row uses the Cartesian product of its learning-rate and weight-decay candidates. Selected settings are listed in the task descriptions; non-SGD denotes Adam, AdamW, RAD, Muon, and RADO.

Task	Optimizer(s)	Learning-rate candidates	Weight-decay candidates
ViT	Adam, RAD, RADO	$\{10^{-4}, 3\times10^{-4}, 10^{-3}\}$	$\{0, 5\times10^{-5}\}$
	AdamW	$\{10^{-4}, 3\times10^{-4}, 10^{-3}\}$	$\{0, 0.03\}$
	Muon	$\{10^{-5}, 3\times10^{-5}, 10^{-4}\}$	$\{0, 5\times10^{-5}\}$
	SGD	$\{0.003, 0.01, 0.03\}$	$\{0, 10^{-4}\}$
Alpaca	Non-SGD	$\{2\times10^{-5}, 5\times10^{-5}, 10^{-4}\}$	$\{0, 0.01\}$
	SGD	$\{0.03, 0.1, 0.3\}$	$\{0, 0.01\}$
WikiText-103	Non-SGD	$\{10^{-4}, 3\times10^{-4}, 10^{-3}\}$	$\{0, 0.01\}$
	SGD	$\{0.1, 0.5, 1.0\}$	$\{0, 10^{-4}\}$

RADO settings and parameter routing. The reported RADO runs use $(\beta_1, \beta_2) = (0.9, 0.999)$, $\zeta = 10^{-16}$, and matrix ratio $r = 90$. The main comparisons use $s_\delta = 10^{-4}$ and $\varepsilon_\delta = 10^{-8}$ without task-specific overrides of these constants. Damping horizons T are 78,200 optimizer updates for ViT, 2000 for Alpaca, 10,000 for WikiText-103, and 5000 for LLaVA. Resuming restores the first and second moments and tensor-local step counters while retaining the original horizon. In ViT, Transformer weight matrices are matrix-eligible; bias, normalization, and task-specific excluded parameters use the standard branch. In both GPT-2 tasks, Transformer-block weight matrices use the matrix branch; token and positional embeddings, final normalization, and bias/norm vectors use the standard branch. The tied token-embedding/LM-head parameter is included once and is not orthogonalized.

ViT–CIFAR100. ViT uses FP32 training (with the specified BF16 conversion inside NS5) and a cosine schedule over 100 epochs, with the task-level warmup and no additional optimizer-specific

gradient clipping. The comparison uses five RADO runs with $\zeta = 10^{-16}$. ViT learning rates are 3×10^{-4} for Adam, AdamW, RAD and RADO, 3×10^{-5} for Muon, and 10^{-2} for SGD. Weight decays are 5×10^{-5} for Adam, RAD and RADO, 0.03 for AdamW, 0 for Muon, and 10^{-4} for SGD. For Muon seed 42, the learning curve uses the last record for each repeated epoch, and score selection uses the maximum logged validation accuracy.

Alpaca. Alpaca uses FP16, gradient-norm clipping at 1.0, and a cosine schedule with 100 warmup steps, ending at step 2000. Learning rates are 5×10^{-5} for the adaptive/matrix optimizers and 0.1 for SGD; weight decay is 0.01 for AdamW and RADO, and 0 for Adam, Muon, RAD and SGD. Evaluation and training losses are matched by step. Training loss here is the last logged training interval, not the mean over the entire run.

WikiText-103. WikiText-103 uses FP16, gradient-norm clipping at 1.0, and a cosine schedule with 500 warmup steps over 10,000 updates. The Muon implementation is `SingleDeviceMuonWithAuxAdam`. The comparison uses final evaluation and whole-run training summaries, without a separate checkpoint-selection step. Learning rates are 3×10^{-4} for Adam, AdamW, RAD, Muon and RADO, and 0.5 for SGD. WD is 0 for Adam and Muon, 0.01 for AdamW, RAD and RADO, and 10^{-4} for SGD. SGD uses Nesterov momentum 0.9; RAD uses $\zeta = 10^{-16}$.

C.3 Ablation Setup and Variant Definitions

The ablations use a 12-layer, 12-head GPT-2 model with hidden size 768 (124,439,808 parameters), trained from scratch on WikiText-103 for 10,000 updates with seed 42. All runs use sequence length 512, effective batch size 64, learning rate 3×10^{-4}, weight decay 0.01, FP16, a cosine schedule with 500 warmup steps, and evaluation every 500 steps. The ablations use $(\beta_1, \beta_2) = (0.9, 0.999)$, $\zeta = 10^{-16}$, $r = 90$, $\varepsilon_\delta = 10^{-8}$, $T = 10{,}000$, and gradient-norm clipping at 1.0. Figure 3 therefore shows one run per variant, without seed averaging or uncertainty bands. Training settings and the remaining optimizer hyperparameters are shared without independent tuning; damping-strength differences are specified below.

For the ablation variants below, let

$$\begin{aligned} H_t &= (V_t \odot \delta_t^2 + \zeta)^{-1/2}, \quad & H_t^{(0)} &= (V_t + \zeta)^{-1/2}, \\ U_t &= M_t \odot H_t, & U_t^{(0)} &= M_t \odot H_t^{(0)}. \end{aligned} \tag{20}$$

Here M_t is the persistent first-moment state (`exp_avg`), H_t is the element-wise preconditioner (`denom`), Φ is `my_ln`, and NS5 is `zeropower_via_newtonschulz5(..., steps=5)`. Each variant's H_t and U_t use its own states and damping strength. Full RADO and the no-compression variant use $s_\delta = 10^{-4}$; both post-NS5 placement variants use $s_\delta = 1$. Static-damping variants bypass this schedule. The directions in Table 5 are shown before the common aspect-ratio and outer learning-rate scalars.

Table 5: Ablation paths and implementation identifiers. The last two rows use $s_\delta = 1$, versus 10^{-4} for full RADO and no compression; static variants set $\delta_t \equiv 1$. Preconditioning uses each row's configured damping.

Paper name	Implementation identifier	Matrix direction $Q_t^{(v)}$
Full RADO	`RADO`	$\mathrm{NS5}(\Phi(U_t))$
Static damping, $\delta_t \equiv 1$	`RADO_no_delta`	$\mathrm{NS5}(\Phi(U_t^{(0)}))$
No compression	`RADO_no_ln`	$\mathrm{NS5}(U_t)$
Static damping without compression	`RADO_no_ln_delta`	$\mathrm{NS5}(U_t^{(0)})$
Compression after NS5	`RADO_orth_ln`	$\Phi(\mathrm{NS5}(U_t))$
Preconditioning after NS5	`RADO_orth_expavg`	$\mathrm{NS5}(\Phi(M_t)) \odot H_t$

Here `exp_avg` denotes M_t, not an operation. Compression after orthogonalization uses $\Phi(\mathrm{NS5}(U_t))$; post-orthogonalization preconditioning instead applies compression and NS5 to M_t

before multiplying by H_t. The first four variants examine damping and compression under a common precondition-first ordering; the remaining two compare alternative placements with a different damping strength.

C.4 LLaVA Protocol and Benchmark Definitions

Experiments use the official LLaVA training framework (Liu et al., 2024a). We sample a fixed 80,000-example subset without replacement from the 665,298-example LLaVA-v1.5 mixture, using seed 20260919. Sampling is proportionally stratified by the top-level image-path prefix (COCO, GQA, OCR-VQA, TextVQA, or Visual Genome), with a separate text-only stratum; selected examples retain their original source order. Every optimizer and seed uses the same subset, keeping the training examples and their source composition fixed across comparisons. LLaVA-v1.5-7B is fine-tuned on this subset for one epoch (5000 steps), with effective batch size 16. LoRA has rank 128, scaling parameter 256, and dropout 0.05. The projector learning rate is 2×10^{-5}. The LoRA parameter-group learning rate is the base learning rate times the LoRA multiplier: 2×10^{-4} for AdamW and 10^{-5} for both Muon and RADO. Including the matrix ratio gives the matrix-step scales in Table 13: 2×10^{-4}, 6.7×10^{-4}, and 9×10^{-4}, respectively. Muon uses weight decay 0 and RADO uses 0.01. AdamW uses weight decay 0.01 for all three seeds. The runs use `evaluation_strategy=no`, so only downstream benchmark evaluation, not a training-time validation-loss trajectory, is available. The SHA256 below identifies the exact UTF-8 bytes of `llava_v1_5_mix80k_fixed_seed20260919.json`, serialized with two-space indentation, unescaped Unicode, and a terminal newline; it does not cover the image files:

`33aa932489266f5a81eb4e13fc3b39d677f2a3e6d6ad59320c059868d6f42b19`

Training uses microbatch 4, gradient accumulation 4, BF16/TF32, maximum length 2048, gradient-norm clipping at 1.0, a cosine schedule, and 3% warmup (150 steps). Only LoRA and multimodal-projector parameters are trainable. The two-dimensional LoRA A/B tensors use the RADO matrix branch, while all projector parameters, including its matrices, use the standard branch. LoRA is not attached to the vision tower, projector, or LM head. Only the base-LR 2×10^{-5} groups are used. LoRA multiplier/ratio pairs are $(10, 1)$ for AdamW, $(0.5, 67)$ for Muon, and $(0.5, 90)$ for RADO. RADO seed 42 resumed from checkpoint 4000 with moments and tensor-local step counters restored and $T = 5000$ unchanged; its whole-run loss uses the complete deduplicated step history, including the steps before recovery. For the pairwise training-loss progress metric, losses are averaged in non-overlapping 100-step blocks and then across three seeds. The crossing is the first block end after which RADO remains no higher than the baseline's final block loss through step 5000.

LLaVA hyperparameter selection. Candidate base/projector learning rates were $\{10^{-5}, 2 \times 10^{-5}, 3 \times 10^{-4}\}$, with LoRA multipliers $\{0.5, 1, 10\}$. Optimizers were tuned separately; not every Cartesian-product combination completed training. Candidates were screened using aligned training-loss trajectories, followed by downstream benchmark evaluation for completed runs. Thus the benchmark suite also informed configuration selection. The selected RADO configuration uses matrix ratio $r = 90$.

Benchmark definitions. POPE (Li et al., 2023) is the unweighted mean of F1 on adversarial (3000), popular (3000), and random (2910) questions. MME (Fu et al., 2025) has 2374 questions over 1187 images; its total is computed within each run by summing perception and cognition before calculating the cross-seed mean and sample standard deviation. MMBench (Liu et al., 2024b) is the local development set (4377 questions), ScienceQA (Lu et al., 2022) uses all 4241 questions rather than the image-only subset, TextVQA (Singh et al., 2019) uses 5000 questions, and GQA (Hudson & Manning, 2019) uses 12578 questions. VQAv2 (Goyal et al., 2017) reports official test-dev results from 107394 predictions per run. Scores are reported on each benchmark's own scale, without an overall cross-benchmark average.

C.5 Resource Measurements and Time-to-Target Estimates

Task-specific time estimates. For each optimizer with a target crossing in Table 3, the Alpaca seconds-per-step value is the mean of `time_per_step_sec` from its five selected runs' `efficiency.json` files. The WikiText-103 value is the mean of `train_runtime`/10,000 from

its five selected complete runs' `train_results.json` files. For Adam, AdamW, Muon, and RADO, the Alpaca means are 0.542177, 0.541674, 0.557039, and 0.585678 seconds per step, respectively. For Adam, AdamW, RAD, and RADO, the WikiText-103 means are 0.417867, 0.419702, 0.424988, and 0.466581 seconds per step, respectively. With the integer crossing step τ_o from Table 3, estimated minutes equal $\tau_o \bar{s}_o/60$, where $\bar{s}_o$ is the task-specific mean seconds per step. Estimated relative reduction is $100(1 - \tau_o \bar{s}_o/(T_{\mathrm{Adam}} \bar{s}_{\mathrm{Adam}}))$. WikiText-103's full-run runtime averages all recorded training work over 10,000 steps. For both tasks, target times are estimated from interpolated seed-mean loss crossings and average step durations, rather than directly timed.

Figure 4: estimated time axis. The figure uses the same five complete WikiText-103 runs per optimizer and the same `train_runtime` field as the estimates above. For run j, let R_j be its measured total training duration and $K_j = 10{,}000$ its maximum logged step. A training-loss record at step k is assigned estimated time $kR_j/(3600K_j)$ hours; no per-step timestamps were retained. The input losses are Trainer averages logged every 10 steps. For each optimizer, its five curves are linearly interpolated onto 300 equally spaced points over their common time interval, then averaged; shading is ± 1 sample standard deviation across seeds, with no additional smoothing. Total runtime includes in-loop data waiting, evaluation every 500 steps, and checkpoint saving every 1000 steps. It excludes pre-training preparation, final best-checkpoint loading, and model saving or evaluation after the training loop. Uniform time allocation spreads these local pauses across the estimated time axis. Unlike Table 3, which multiplies an evaluation-loss crossing by the mean seconds per step, this figure averages training-loss curves after per-run time rescaling. The LLaVA resource probe below measures a separate workload.

LLaVA resource probe. AdamW, Muon, and RADO were run sequentially on one NVIDIA GeForce RTX 5090 with 60-second cooldowns, using the same fixed subset and data order (seed 42), microbatch 4, four accumulation steps, maximum length 2048, BF16, and gradient checkpointing. After 30 warm-up steps, 300 full training steps were timed with CUDA synchronization; two additional steps were profiled. The training-step measurement includes forward, backward, and optimizer work but excludes model loading, checkpoint saving, and downstream evaluation. In AdamW, Muon, RADO order, training-step times are 5.333 ± 0.596, 5.534 ± 0.595, and 5.623 ± 0.597 seconds; optimizer-state tensors occupy 1.363, 0.724, and 1.363 decimal GB; profiled optimizer GPU times are 0.011, 0.384, and 0.541 seconds per step; and observed peak device memory is 22.89, 21.92, and 22.50 GiB. The training-step standard deviations describe variation within one run per optimizer. Optimizer GPU time is the CUDA time attributed to `Optimizer.step` during two profiled steps, not an isolated synchronized wall-clock measurement. Optimizer-state memory sums persistent state tensors and excludes temporary workspaces. Peak GPU memory comes from whole-device samples at 0.5-second intervals and cannot isolate optimizer allocations or guarantee capture of shorter peaks.

Reproduction material. The configuration fields above accompany the full aggregated results and learning curves in Appendix D.

D Full Results and Additional Learning Curves

This appendix gives the task-level metrics behind Tables 1 and 3, together with additional training curves. For ViT, Alpaca, and WikiText-103, lines show five-seed means and bands show ± 1 sample standard deviation. LLaVA uses three seeds and the block averaging specified below. Configuration differences and selection rules appear in Appendix C.

D.1 ViT–CIFAR100

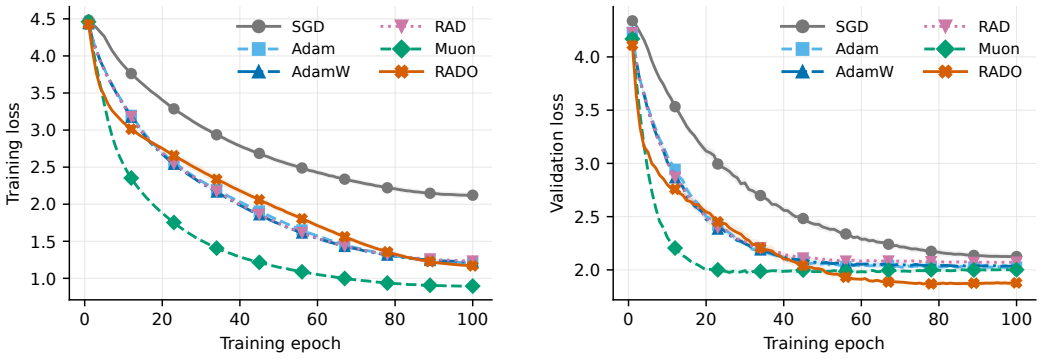
Table 6 reports peak validation accuracy and validation loss at each seed's selected epoch. RADO has the highest mean accuracy among the methods in this table; the reported loss uses the same selected records. Figure 5 shows another distinction: Muon reaches Adam's final validation-loss target first, at epoch 19 (Table 7). SGD, AdamW, and RAD do not reach it within the recorded budget.

Table 6: ViT–CIFAR100 results over five seeds (mean $\pm$ sample standard deviation). Each seed is selected by its maximum validation accuracy; loss and epoch are taken from the same record. Bold marks the best mean in each performance metric.

Optimizer	Peak val. acc. (%) ↑	Val. loss ↓	Selected epoch
SGD	59.53 ± 0.74	2.1255 ± 0.0289	97.2 ± 1.9
Adam	65.01 ± 0.54	2.0232 ± 0.0222	98.6 ± 1.5
AdamW	64.44 ± 0.48	2.0300 ± 0.0232	94.4 ± 5.6
RAD	63.49 ± 0.52	2.0688 ± 0.0180	94.8 ± 4.1
Muon	68.04 ± 0.43	2.0002 ± 0.0141	97.6 ± 1.8
RADO	$\mathbf{68.97 \pm 0.47}$	$\mathbf{1.8714 \pm 0.0233}$	95.4 ± 3.2

Table 7: ViT training-epoch reduction relative to Adam's final seed-mean validation loss (2.026400). Reduction is $100(100-\tau_o)/100$; – means that the target was not reached. Adam's row is the reference budget endpoint; other attained entries use the first recorded epoch at or below the target.

Optimizer	Epoch to target ↓	Reduction (%) ↑
SGD	–	–
Adam	100	0.00
AdamW	–	–
RAD	–	–
Muon	19	**81.00**
RADO	47	53.00



(a) Training loss (b) Validation loss

Figure 5: ViT–CIFAR100 loss trajectories across the 100-epoch budget. Lines and bands summarize five seeds.

D.2 Alpaca Instruction Tuning

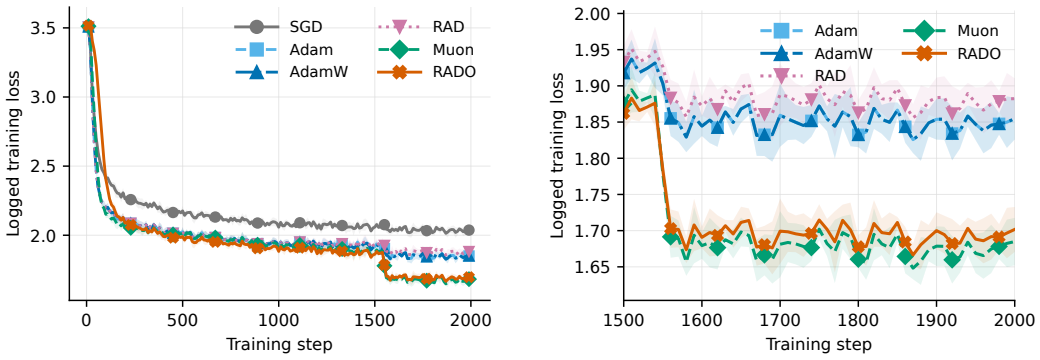
Table 8 aggregates full-precision final records. Training loss is the last logged interval at step 2000, whereas evaluation loss and perplexity are the final evaluation. RADO's lower evaluation perplexity coexists with higher training loss than Muon. Figure 6 complements the evaluation curves in Figure 2. Table 9 gives the loss target, interpolated crossing steps, and methods that do not attain it.

Table 8: GPT-2 Alpaca instruction-tuning results at the common final step 2000 over five seeds (mean $\pm$ sample standard deviation). Train loss is the logged interval loss at that step. Bold marks the best mean in each metric.

Optimizer	Train loss ↓	Eval. loss ↓	PPL ↓
SGD	2.0407 ± 0.0267	1.9577 ± 0.0138	7.083 ± 0.098
Adam	1.8547 ± 0.0288	1.8371 ± 0.0134	6.279 ± 0.084
AdamW	1.8547 ± 0.0288	1.8371 ± 0.0134	6.279 ± 0.084
RAD	1.8823 ± 0.0282	1.8525 ± 0.0130	6.376 ± 0.083
Muon	$\mathbf{1.6849 \pm 0.0328}$	1.8098 ± 0.0131	6.110 ± 0.080
RADO	1.7017 ± 0.0310	$\mathbf{1.7945 \pm 0.0135}$	$\mathbf{6.017 \pm 0.081}$

Table 9: Alpaca training-step reduction relative to Adam's final seed-mean evaluation loss (1.837102). Reduction is $100(2000 - \tau_o)/2000$; – means that the target was not reached. Adam's row is the reference budget endpoint. Other attained entries are first crossings, linearly interpolated between adjacent seed-mean evaluation-loss records and rounded to the nearest integer step; reductions use the displayed integers.

Optimizer	Step to target ↓	Reduction (%) ↑
SGD	–	–
Adam	2000	0.00
AdamW	1916	4.20
RAD	–	–
Muon	1213	39.35
RADO	990	**50.50**



(a) Full trajectory (b) Final 500 steps

Figure 6: Alpaca training loss over five seeds. The full trajectory includes SGD; the late view omits it to resolve the remaining curves.

D.3 WikiText-103 Pretraining

Table 10 reports whole-run training loss alongside final evaluation loss and perplexity. RADO has the lowest mean final evaluation loss and perplexity among the methods in this table. Figure 7 shows

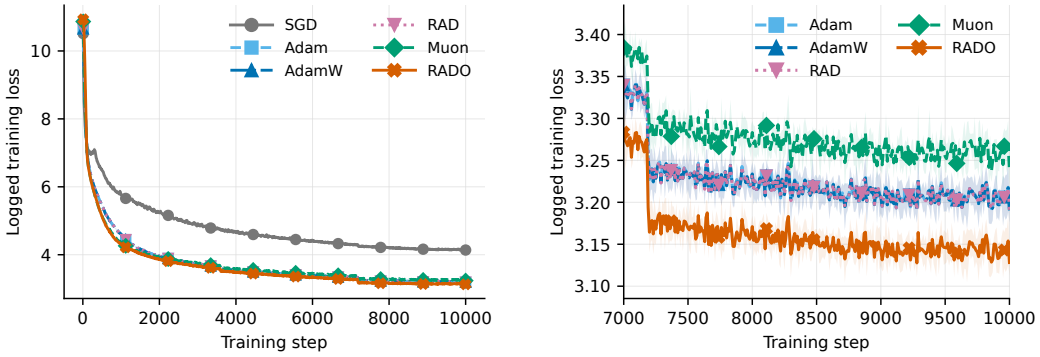
the training trajectories; Figure 2 gives the corresponding evaluation histories. RADO reaches the Adam-relative target at interpolated step 6740, a 32.60% reduction (Table 11).

Table 10: GPT-2 pretraining on WikiText-103 at 10,000 optimizer steps over five seeds (mean ± sample standard deviation). Bold marks the best mean in each metric.

Optimizer	Whole-run train loss ↓	Eval. loss ↓	PPL ↓
SGD	4.7857 ± 0.0084	4.0268 ± 0.0082	56.085 ± 0.458
Adam	3.7332 ± 0.0014	3.1941 ± 0.0021	24.388 ± 0.051
AdamW	3.7332 ± 0.0011	3.1930 ± 0.0017	24.361 ± 0.042
RAD	3.7332 ± 0.0010	3.1934 ± 0.0018	24.371 ± 0.043
Muon	3.7370 ± 0.0009	3.2025 ± 0.0007	24.593 ± 0.018
RADO	$\mathbf{3.6548 \pm 0.0011}$	$\mathbf{3.1250 \pm 0.0013}$	$\mathbf{22.760 \pm 0.030}$

Table 11: GPT-2 pretraining-step reduction relative to Adam's final seed-mean evaluation loss (3.194082). Reduction is $100(10000 - \tau_o)/10000$; – means that the target was not reached. Adam's row is the reference budget endpoint. Other attained entries are first crossings, linearly interpolated between adjacent seed-mean evaluation-loss records and rounded to the nearest integer step; reductions use the displayed integers.

Optimizer	Step to target ↓	Reduction (%) ↑
SGD	–	–
Adam	10000	0.00
AdamW	9411	5.89
RAD	9481	5.19
Muon	–	–
RADO	6740	**32.60**



(a) Full trajectory

(b) Final 3000 steps

Figure 7: WikiText-103 training loss over five seeds. The full view includes SGD; the late view omits it.

D.4 LLaVA Fine-Tuning and Training Progress

Table 13 gives the selected configurations and whole-run training loss. Each point in Figure 8 first averages a non-overlapping block of 100 steps within a seed, then reports the mean and sample standard deviation over three seeds. The seven reported downstream benchmarks appear in Table 2; their evaluation subsets are specified in Appendix C.

Table 12 separates the two components of the MME benchmark. RADO has the highest mean perception score, while Muon has the highest mean cognition score; these components are not counted as additional benchmarks in the seven-benchmark summary.

Table 12: MME component scores (three seeds, mean $\pm$ sample standard deviation). Per-run component scores sum to the MME score in Table 2. Bold marks the best mean per component.

Optimizer	Perception ↑	Cognition ↑
AdamW	1408.22 ± 20.76	302.62 ± 15.91
Muon	1441.02 ± 22.93	$\mathbf{314.29 \pm 20.44}$
RADO	$\mathbf{1491.53 \pm 12.88}$	287.50 ± 9.11

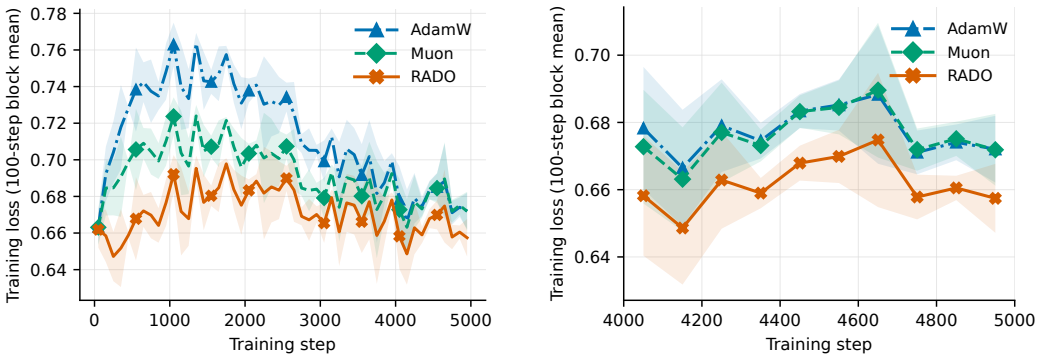
Table 13: LLaVA-v1.5-7B training configurations and results over seeds 42, 123, and 1234 (mean $\pm$ sample standard deviation). Loss is averaged over all 5000 unique steps per seed. AdamW uses WD=0.01 for all three seeds. The LoRA parameter-group learning rate equals the base learning rate times the LoRA multiplier; for matrix-branch parameters, the matrix-step scale additionally includes the matrix ratio r.

Optimizer	Base LR	LoRA mult.	Ratio	Matrix-step scale	WD	Train loss ↓
AdamW	2×10^{-5}	10	1	2×10^{-4}	0.01	0.7133 ± 0.0002
Muon	2×10^{-5}	0.5	67	6.7×10^{-4}	0	0.6925 ± 0.0002
RADO	2×10^{-5}	0.5	90	9×10^{-4}	0.01	$\mathbf{0.6715 \pm 0.0001}$

This comparison uses AdamW and Muon as pairwise references and training loss as the progress metric; the runs have no training-time validation-loss trajectory. Each target is the baseline's final three-seed mean block loss. Because the non-monotone curves begin below their final values, we report the first block end after which RADO remains at or below the target through step 5000. Both targets give step 4800 and a 4.00% reduction (Table 14). This sustained training-loss criterion is distinct from the interpolated evaluation-loss crossing used in Table 3.

Table 14: LLaVA pairwise training-step efficiency of RADO. For each baseline, the target is its final three-seed mean training loss over a 100-step block. Because these training-loss curves are non-monotone, τ is the first block end after which RADO remains at or below the target through step 5000. Reduction is $100(5000 - \tau)/5000$. This is a training-loss proxy, not a validation-loss or wall-clock comparison.

Baseline	Final-loss target	RADO τ	Reduction (%) ↑
AdamW	0.672050	4800	4.00
Muon	0.671869	4800	4.00



(a) Full trajectory (b) Final 1000 steps

Figure 8: LLaVA training loss after averaging non-overlapping 100-step blocks. Lines and bands summarize the three seeds. The full and late views use the same averaging rule.