

Variance-Adaptive Muon: Pre-Orthogonalization Variance Modulation for Efficient Language Model Pretraining

Jingru Li, Yibo Fan, Huan Li*

College of Artificial Intelligence, Nankai University, Tianjin, China
{jingru_lee,yibofan}@mail.nankai.edu.cn, lihuanss@nankai.edu.cn

Abstract

Optimizer design plays a central role in efficient language model pretraining, directly affecting optimization dynamics, convergence speed, and compute cost under fixed training budgets. Muon has emerged as a strong optimizer by orthogonalizing momentum updates, yielding a matrix-valued analogue of sign-based normalization. However, unlike Adam-style methods, Muon does not explicitly incorporate gradient-variance information into its updates. Motivated by Adam’s variance-adaptive interpretation, we propose Muon-NSR and Muon-VS, two variance-adaptive Muon variants for language model pretraining. Muon-NSR applies noise-to-signal ratio (NSR) modulation before Newton–Schulz orthogonalization, whereas Muon-VS uses variance scaling (VS) without introducing any additional hyperparameters beyond those of Muon. Both methods preserve Muon’s spectral normalization structure while requiring only one additional variance buffer. Experiments on Llama-style and GPT-2 pretraining across model scales from 125M to 1.2B parameters show that our methods improve over well-tuned Muon baselines and remain competitive with representative adaptive Muon-family baselines. On Llama-1.2B, Muon-VS achieves a $1.33\times$ step-to-target speedup over a well-tuned Muon baseline, with Muon’s final validation loss as the target. These results indicate that variance-adaptive modulation is a simple and effective mechanism for improving Muon-style optimizers in language model pretraining.

1 Introduction

Large language models (LLMs) are commonly developed through self-supervised pretraining on large-scale text corpora, a paradigm that has become central to modern natural language processing (NLP) (Raffel et al., 2020; Brown et al., 2020).

*Corresponding author.

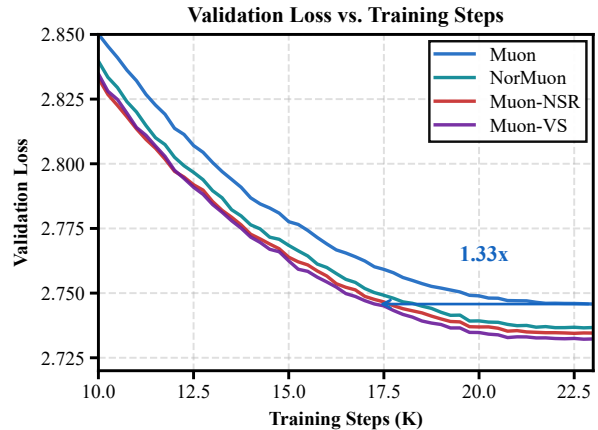


Figure 1: **Suite A Llama-1.2B pretraining.** Validation loss on the C4-EN validation split during pretraining on DCLM with the $1\times$ Chinchilla token budget.

Scaling studies show that this paradigm incurs substantial computational cost that grows jointly with model size and the total number of training tokens (Kaplan et al., 2020; Hoffmann et al., 2022). Under a fixed architecture and data budget, optimizer design provides a direct algorithmic lever for improving pretraining efficiency by reducing the number of steps or wall-clock time required to reach a target validation loss (Wen et al., 2026; Semenov et al., 2025; Shah et al., 2025).

Adam (Kingma and Ba, 2015) and its variants, particularly AdamW (Loshchilov and Hutter, 2019), remain standard choices for Transformer-based language model pretraining (Wen et al., 2026; Liang et al., 2026). Recent analysis shows that when the first- and second-moment exponential moving averages (EMAs) use the same decay rate, Adam can be interpreted as a variance-adaptive sign update that attenuates high-variance coordinates (Orvieto and Gower, 2025). Under this interpretation, Adam’s per-coordinate update decomposes into two complementary mechanisms: *sign-based coordinate-wise normalization* and *variance-based noise-aware damping*.

Muon (Jordan et al., 2024) extends sign normalization to matrix-valued momentum updates by approximating the orthogonal polar factor of a momentum matrix with Newton–Schulz iterations, yielding a spectrally normalized analogue of a sign update. This orthogonalized update has made Muon competitive in language model pretraining and optimizer benchmarks (Wen et al., 2026; Liu et al., 2025; Shah et al., 2025), and industry-scale reports further associate Muon with faster convergence and improved training stability (DeepSeek-AI, 2026). Despite these empirical strengths, standard Muon captures the spectral analogue of sign normalization but lacks online variance statistics and thus explicit variance-aware damping of high-variance or noise-dominated update directions. This mechanistic gap therefore motivates the central question.

Can Adam-style variance-adaptive modulation be incorporated into Muon’s matrix-valued sign updates for efficient language model pretraining?

Contributions. To address this question while preserving Muon’s spectral normalization, we propose **Muon-NSR** and **Muon-VS**, two variance-adaptive Muon variants for efficient language model pretraining. Our contributions are twofold:

- **Variance-adaptive Muon updates.** Using online centered gradient-variance estimates, Muon-NSR attenuates high-uncertainty update entries via noise-to-signal ratio (NSR) modulation, while Muon-VS applies variance scaling without introducing additional optimizer-specific hyperparameters. Both variants modulate the update proxy before Newton–Schulz orthogonalization, thereby preserving Muon’s spectral update structure with only one additional variance buffer.
- **Matched pretraining evaluation.** We evaluate Muon-NSR and Muon-VS across Llama-style and GPT-2 pretraining regimes spanning model scales from 125M to 1.2B parameters. Under matched training recipes and fixed pretraining budgets, both variants converge faster and achieve lower validation loss than the evaluated AdamW and Muon-family baselines. On Llama-1.2B under the Suite A protocol (Wen et al., 2026), Muon-VS achieves a $1.33\times$ step-to-target speedup over Muon, using Muon’s final validation loss as the target.

2 Related Work

Stochastic gradient descent (Robbins and Monro, 1951) and its momentum variants (Polyak, 1964) are classical first-order optimization methods. Sign-based coordinate-wise normalization discards update magnitudes while preserving directional information. signSGD (Bernstein et al., 2018) applies this normalization to stochastic gradients, while Signum (Bernstein et al., 2018) and Lion (Chen et al., 2023) extend it to momentum directions.

Beyond sign-based normalization, adaptive gradient methods rescale coordinates using gradient statistics, including AdaGrad (Duchi et al., 2011) and RMSProp (Tieleman and Hinton, 2012). Adam (Kingma and Ba, 2015) combines momentum with second-moment adaptivity. Moreover, under tied first- and second-moment EMA decay rates, Adam can be interpreted as a sign update with variance-based damping (Orvieto and Gower, 2025). AdamW (Loshchilov and Hutter, 2019) further decouples weight decay from Adam’s update.

Coordinate-wise adaptive methods commonly view model parameters as a vector and apply diagonal rescaling per coordinate, without explicitly exploiting layer-wise parameter-block structures such as vectors and matrices (Lau et al., 2025). K-FAC (Martens and Grosse, 2015) approximates natural-gradient updates using Kronecker-factored curvature, whereas Shampoo (Gupta et al., 2018) and SOAP (Vyas et al., 2025) use structured matrix preconditioning. Muon (Jordan et al., 2024) instead approximates the orthogonal polar factor of a momentum matrix with Newton–Schulz iterations, yielding a matrix-valued analogue of sign-based coordinate-wise normalization that maps nonzero singular values approximately to one.

Muon’s success has motivated extensions incorporating adaptive normalization into its spectral-update framework. AdaMuon (Si et al., 2025) combines element-wise second-moment normalization after orthogonalization with sign-stabilized orthogonal updates. NorMuon (Li et al., 2026) applies row-wise normalization to orthogonalized updates using neuron-wise second moments. Muon² (Liu et al., 2026) preconditions Muon’s momentum using Adam-style uncentered second moments before Newton–Schulz orthogonalization. By contrast, we use centered variance estimates, rather than uncentered second moments or post-orthogonalization statistics, to introduce Adam-inspired noise-aware damping before orthogonalization.

3 Motivation from the Variance-Adaptive Perspective of Adam

An equivalent form of Adam with equal betas.

Consider a scalar coordinate $x_t \in \mathbb{R}$ at iteration t . Let $g_t \in \mathbb{R}$ be the stochastic gradient sample, m_t and v_t be the EMAs of the first and second moments, respectively, $\beta_1, \beta_2 \in (0, 1)$ be their decay rates, $\eta_t > 0$ be the learning rate, and d_t be the coordinate-wise preconditioned update direction. For matrix- or vector-valued quantities, unless stated otherwise, powers, square roots, and divisions are understood element-wise. Omitting bias correction and the stability term ε for analytical tractability, the Adam update can be written as

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \quad (1)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2, \quad (2)$$

$$d_t = \frac{m_t}{\sqrt{v_t}} = \frac{m_t}{\sqrt{m_t^2 + v_t - m_t^2}}, \quad (3)$$

$$x_{t+1} = x_t - \eta_t d_t.$$

Recently, [Orvieto and Gower \(2025\)](#) observed that employing a shared EMA decay rate for both first- and second-moment estimators, denoted as $\beta_1 = \beta_2 = \beta$, maintains competitive performance across various settings. Under this shared parameterization, [Orvieto and Gower \(2025, Proposition 1\)](#) proved that $v_t - m_t^2$ satisfies the recursion

$$v_t - m_t^2 = \beta (v_{t-1} - m_{t-1}^2) + \beta(1 - \beta) (m_{t-1} - g_t)^2. \quad (4)$$

Replacing (2) with (4) yields an equivalent form.

Interpretation of Adam as a variance-adaptive sign update.

Assume that the stochastic gradient g_t at iteration t is generated from a Gaussian distribution with unknown and slowly varying mean and variance, $g_t \sim \mathcal{N}(\mu, \sigma^2)$. Let $(\mu_{t-1}, \sigma_{t-1}^2)$ denote the estimate of the gradient's mean and variance from the previous iteration. Upon observing g_t , the estimate (μ_t, σ_t^2) is defined as the solution of the following regularized maximum likelihood estimation problem ([Orvieto and Gower, 2025](#)):

$$(\mu_t, \sigma_t^2) = \arg \min_{\mu \in \mathbb{R}, \sigma^2 > 0} \left\{ -\log p(g_t | \mu, \sigma^2) + \frac{1}{\lambda} \text{KL}(\mathcal{N}(\mu_{t-1}, \sigma_{t-1}^2) \parallel \mathcal{N}(\mu, \sigma^2)) \right\}, \quad (5)$$

where $\lambda > 0$ controls the trade-off between fitting the current observation and penalizing deviation

from the previous estimate. The standard closed forms of the Gaussian negative log-likelihood and the Gaussian–Gaussian Kullback–Leibler divergence used here are omitted for brevity and follow those in [Orvieto and Gower \(2025\)](#).

With the identification $\beta = \frac{1}{1+\lambda} \in (0, 1)$, [Orvieto and Gower \(2025, Theorem 4.1\)](#) proved that the minimizer of (5) admits a closed-form solution that is recursively defined as follows:

$$\mu_t = \beta \mu_{t-1} + (1 - \beta) g_t, \quad (6)$$

$$\sigma_t^2 = \beta \sigma_{t-1}^2 + \beta(1 - \beta) (\mu_{t-1} - g_t)^2. \quad (7)$$

Comparing (4) with (7) and (1) with (6) reveals that $\sigma_t^2 = v_t - m_t^2$ and $\mu_t = m_t$, implying that m_t and $v_t - m_t^2$ in Adam's updates serve as online estimators of the local gradient mean and variance, respectively. Substituting these identities into (3) yields the variance-adaptive update

$$d_t = \frac{m_t}{\sqrt{m_t^2 + \sigma_t^2}} = \frac{\text{sign}(m_t)}{\sqrt{1 + \sigma_t^2/m_t^2}}, \quad (m_t \neq 0), \quad (8)$$

where $\text{NSR}_t = \sigma_t/|m_t|$ denotes the estimated local noise-to-signal ratio. Under the shared-decay interpretation, the update in (8) decomposes Adam's per-coordinate update into two coupled mechanisms. The term $\text{sign}(m_t)$ corresponds to *sign-based coordinate-wise normalization* ([Bernstein et al., 2018](#)). The factor $(1 + \text{NSR}_t^2)^{-1/2}$ corresponds to *variance-based noise-aware damping*, suppressing step sizes in noise-dominated regimes ([Orvieto and Gower, 2025](#)).

Muon's Spectral Sign Update. Muon ([Jordan et al., 2024](#)) applies the matrix sign map, defined as the orthogonal polar factor, to a Nesterov-style momentum update proxy. This operation generalizes sign-based normalization from scalar coordinates to matrices by replacing coordinate-wise sign normalization with spectral normalization. For a matrix-valued parameter block, let $G_t \in \mathbb{R}^{m \times n}$ denote its stochastic gradient at iteration t . Formally, Muon forms an orthogonalized update direction as

$$\begin{aligned} M_t &= \beta M_{t-1} + G_t, \\ \widetilde{M}_t &= \beta M_t + G_t, \\ O_t &= \text{msign}(\widetilde{M}_t) \in \mathbb{R}^{m \times n}, \end{aligned} \quad (9)$$

where $\beta \in [0, 1)$ is the momentum coefficient, and the operator $\text{msign}(\cdot)$ returns the orthogonal polar factor of a matrix. For a full-rank matrix X with

thin singular value decomposition (SVD) $X = U\Sigma V^\top$, this factor is

$$\text{msign}(X) = UV^\top.$$

Equivalently, $\text{msign}(\cdot)$ maps the nonzero singular values of X to one, flattening the spectrum while retaining the singular vectors. Rather than computing this SVD explicitly, Muon approximates this direction with the K -step Newton–Schulz operator:

$$O_t = \text{NS}_K(\widetilde{M}_t) \approx \text{msign}(\widetilde{M}_t), \quad (10)$$

where $\widetilde{M}_t$ is the momentum-based update proxy and O_t is the Muon update direction.

Motivation for Variance-Adaptive Muon. The equal- β reformulation in (8) decomposes Adam’s update into sign-based coordinate-wise normalization and variance-based noise-aware damping. Muon extends the sign-normalization component to matrix-valued momentum via spectral normalization, but lacks variance estimates, leaving its update proxy without explicit variance-aware attenuation. This gap motivates incorporating Adam-style variance modulation into Muon-style updates while preserving Muon’s spectral normalization.

4 Muon-NSR and Muon-VS

Motivated by the analysis, we introduce **Muon-NSR** and **Muon-VS**, two variance-adaptive extensions of Muon. Both methods apply variance-based modulation before Muon’s spectral normalization step, preserving Muon’s orthogonalized matrix-update structure while incorporating Adam-style adaptivity. Algorithm 1 presents the procedure.

Momentum and Variance. At iteration t , we maintain two coupled exponential moving statistics, following the derivation in Section 3. The momentum estimator M_t is updated according to (6), yielding a smoothed estimate of the local gradient mean under the shared EMA decay rate and serving as the directional signal for the subsequent polar-factor computation. In parallel, the variance surrogate Γ_t is updated by (7). This recursion is driven by the element-wise squared deviation $(G_t - M_{t-1})^2$, which quantifies the discrepancy between the current stochastic gradient and the previous mean estimate. Consequently, Γ_t estimates dispersion around the evolving mean, rather than tracking the raw second moment as in Adam. Together, (M_t, Γ_t) provide the mean and variance information used for variance-adaptive normalization before applying the matrix-valued sign operator.

Algorithm 1 Muon-NSR and Muon-VS.

Require: Initial parameter $W_0 \in \mathbb{R}^{m \times n}$; learning-rate schedule $\{\eta_t\}_{t \geq 1}$; weight decay λ_{wd} ; shared EMA rate β ; NSR coefficient γ used only when $\tau = \text{NSR}$; stabilizer ε ; update scale s_{scale} ; Newton–Schulz operator $\text{NS}_K(\cdot)$; variant selector $\tau \in \{\text{NSR}, \text{VS}\}$.

```

1:  $M_0 \leftarrow 0, \quad \Gamma_0 \leftarrow 0$ 
2: for  $t = 1, 2, \dots$  do
3:    $G_t \leftarrow \nabla_W \mathcal{L}(W_{t-1})$ 
4:    $M_t \leftarrow \beta M_{t-1} + (1 - \beta)G_t$ 
5:    $\Gamma_t \leftarrow \beta \Gamma_{t-1} + \beta(1 - \beta)(M_{t-1} - G_t)^2$ 
6:    $\widehat{M}_t \leftarrow M_t / (1 - \beta^t), \quad \widehat{\Gamma}_t \leftarrow \Gamma_t / (1 - \beta^t)$ 
7:    $\widetilde{M}_t \leftarrow G_t + \frac{\beta}{1 - \beta} \widehat{M}_t$ 
8:   if  $\tau = \text{NSR}$  then
9:      $M_{\tau,t} \leftarrow \frac{\widetilde{M}_t}{\sqrt{\widetilde{M}_t^2 + \gamma \widehat{\Gamma}_t + \varepsilon}}$ 
10:  else if  $\tau = \text{VS}$  then
11:     $M_{\tau,t} \leftarrow \frac{\widetilde{M}_t}{\sqrt{\widehat{\Gamma}_t + \varepsilon}}$ 
12:  end if
13:   $O_t \leftarrow \text{NS}_K(M_{\tau,t})$ 
14:   $W_t \leftarrow W_{t-1}(1 - \eta_t \lambda_{\text{wd}}) - \eta_t s_{\text{scale}} O_t$ 
15: end for
```

Nesterov-style acceleration. Following Muon’s Nesterov update in (9), we form a lookahead proxy from the current gradient G_t and the EMA momentum M_t . Muon uses an unnormalized momentum recursion, whereas our formulation parameterizes the first-moment estimate as an Adam-style EMA. Ignoring the bias correction in line 6 of Algorithm 1, rescaling the EMA momentum in line 4 by $(1 - \beta)^{-1}$ yields

$$\frac{M_t}{1 - \beta} = \beta \frac{M_{t-1}}{1 - \beta} + G_t.$$

After this rescaling, the EMA momentum satisfies the same recursion as Muon’s unnormalized momentum buffer. The proxy in line 7 can therefore be expressed as

$$\widetilde{M}_t = G_t + \beta \frac{M_t}{1 - \beta},$$

which coincides with Muon’s Nesterov-style proxy under the no-bias-correction reparameterization above. Thus, the factor $(1 - \beta)^{-1}$ reparameterizes the Adam-style EMA momentum into Muon’s unnormalized momentum form, rather than acting as an update-scale heuristic. The bias correction in line 6 compensates for the early-step underestimation induced by zero initialization.

Muon-NSR. Muon-NSR normalizes the Muon update proxy by incorporating the variance estimate $\widehat{\Gamma}_t$ into $\widetilde{M}_t$, following the variance-adaptive form in Eq. (8). The normalized proxy is defined as

$$M_{\text{NSR},t} = \frac{\widetilde{M}_t}{\sqrt{\widetilde{M}_t^2 + \gamma \widehat{\Gamma}_t + \varepsilon}}, \quad (11)$$

where $\gamma \geq 0$ controls the strength of variance-based damping and $\varepsilon > 0$ ensures numerical stability. Ignoring the stabilizer ε , each nonzero entry of $\widetilde{M}_t$ reduces to

$$M_{\text{NSR},t} \approx \frac{\text{sign}(\widetilde{M}_t)}{\sqrt{1 + \gamma \widehat{\Gamma}_t / \widetilde{M}_t^2}},$$

which shows that Muon-NSR acts as a soft gate controlled by the estimated noise-to-signal ratio. Entries with large estimated uncertainty relative to their signal strength are therefore damped in the pre-orthogonalization proxy. Because this modulation is applied before Newton–Schulz orthogonalization, the adapted proxy is subsequently mapped into Muon’s approximate polar-factor geometry. Appendix C.2 provides a singular-value analysis of this ordering, while Section 5.3 tests it empirically.

Muon-VS. Muon-VS directly normalizes the update proxy using the estimated gradient variance:

$$M_{\text{VS},t} = \frac{\widetilde{M}_t}{\sqrt{\widehat{\Gamma}_t + \varepsilon}}.$$

Compared with Muon-NSR, this formulation removes the squared proxy term $\widetilde{M}_t^2$ and the sensitivity coefficient γ from the denominator. This simplification reduces the optimizer-specific hyperparameter search space while preserving direct variance-based reweighting of the update proxy. Ignoring the stabilizer ε , Muon-VS can be viewed as the large- γ limiting case of Muon-NSR when $\gamma \widehat{\Gamma}_t \gg \widetilde{M}_t^2$, up to a positive scalar absorbed by the subsequent spectral normalization.

Orthogonalization and Parameter Update. Let $M_{\tau,t} \in \{M_{\text{NSR},t}, M_{\text{VS},t}\}$ denote the variance-modulated update proxy determined by the optimizer type τ . Following the Newton–Schulz approximation in (10), we compute the orthogonalized direction as

$$O_t = \text{NS}_K(M_{\tau,t}).$$

Let $W_t \in \mathbb{R}^{m \times n}$ denote the parameter matrix after the t -th update. The parameter matrix is then updated using Muon’s decoupled weight-decay rule:

$$W_t = W_{t-1}(1 - \eta_t \lambda_{\text{wd}}) - \eta_t s_{\text{scale}} O_t,$$

where η_t is the learning rate and $\lambda_{\text{wd}} \geq 0$ is the weight-decay coefficient. Since the norm of an orthogonalized matrix depends on its shape, practical Muon implementations usually apply a dimension-dependent scaling factor, such as

$$s_{\text{scale}} \in \left\{ 0.2 \sqrt{\max(m, n)}, \sqrt{\max\left(1, \frac{m}{n}\right)} \right\}.$$

These rules compensate for shape-dependent update magnitudes while preserving the spectral normalization induced by $\text{NS}_K(\cdot)$. For a fair comparison, we adopt the same s_{scale} rule as the corresponding Muon baseline in our experiments, thereby more cleanly isolating the effect of pre-orthogonalization variance modulation.

Computational and Memory Cost. For an $m \times n$ parameter matrix, Muon-NSR and Muon-VS add an $O(mn)$ element-wise variance modulation step. For typical matrix-shaped Transformer parameter groups, this added cost is dominated in practice by the $O(Kmn \min(m, n))$ cost of the K Newton–Schulz iterations. Both variants require one additional element-wise variance buffer and therefore have optimizer-state footprints comparable to those of Adam-type optimizers. We report runtime and memory measurements in Appendix D.

5 Experiments

We evaluate Muon-NSR and Muon-VS under fixed pretraining budgets across Llama-style and GPT-2 pretraining settings. To assess optimization efficiency, we report fixed-budget final validation loss and step-to-target and time-to-target speedups using the reference baseline’s final validation loss as the target. Appendix A provides the experimental settings, metric definitions, and implementation details, while Appendix B presents additional results, including final validation losses and perplexities and, for the Llama-style settings, step-to-target and time-to-target speedups at five shared validation-loss targets spanning 20%–100% of Muon’s training budget. To assess whether the pretraining gains extend beyond validation loss, Appendix E provides a compact zero-shot downstream sanity check on the Suite A Llama checkpoints. The source code is publicly available at <https://github.com/jingru-lee/Variance-Adaptive-Muon-EMNLP>.

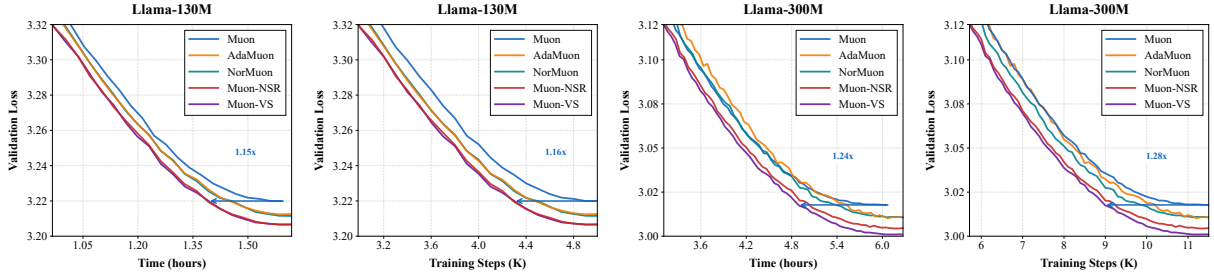


Figure 2: **Suite A Llama pretraining.** Validation loss for **Llama-130M** and **Llama-300M** under the Suite A protocol, plotted against wall-clock time and training steps. The **Llama-1.2B** result is reported in Figure 1.

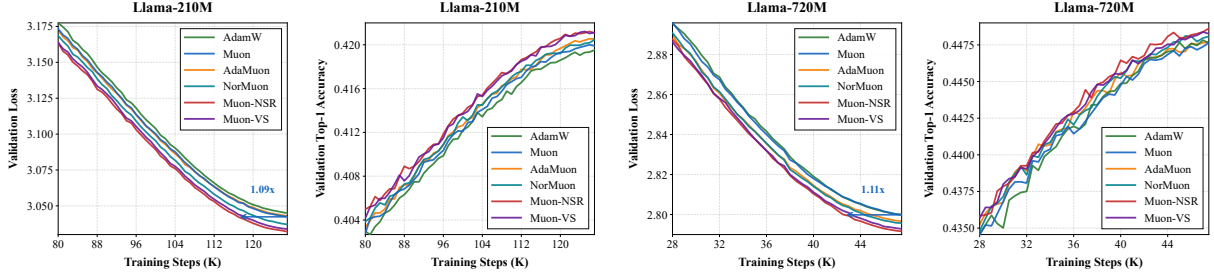


Figure 3: **Suite B Llama pretraining.** Validation loss and validation-set next-token top-1 accuracy for **Llama-210M** and **Llama-720M** under the Suite B protocol.

5.1 Llama Pretraining

To assess whether the proposed methods generalize across Llama-style pretraining regimes, we conduct two complementary suites based on established training recipes. Suite A follows the pre-training optimizer protocol of Wen et al. (2026), while Suite B follows the optimizer benchmarking setting of Semenov et al. (2025). Both protocols include *extensive hyperparameter tuning* and *broad comparisons against more than ten optimizers*, providing rigorous and competitive evaluation settings.

Evaluation Scope. Rather than repeating a broad optimizer ranking, we build on the sweeps of Wen et al. (2026) and Semenov et al. (2025), which position Muon among the strongest pretraining optimizers under tuned protocols. Accordingly, we evaluate whether variance-adaptive modulation improves Muon and compare our variants against Muon-family baselines under matched recipes.

Hyperparameter Reuse. Across all Llama experiments, Muon-NSR and Muon-VS inherit the corresponding protocol and Muon-tuned hyperparameters, with only one additional coarse coefficient γ for Muon-NSR and no optimizer-specific hyperparameter beyond Muon for Muon-VS. This design evaluates whether the proposed variants can serve as drop-in alternatives to Muon-style optimizers with limited additional hyperparameter search.

Suite A Setup. Following the Llama pretraining protocol of Wen et al. (2026), we evaluate Muon-NSR and Muon-VS under a $1\times$ Chinchilla token budget on Llama-130M, Llama-300M, and Llama-1.2B using tuned Muon hyperparameters and the protocol’s warmup–stable–decay (WSD) learning-rate schedule. NorMuon is evaluated at all three scales using the Muon-tuned recipe, while AdaMuon is additionally included for Llama-130M and Llama-300M using the scaling rule selected by the learning-rate and scaling-rule search on Llama-130M in Appendix B.5, with a scale-adjusted learning rate for Llama-300M. All runs use the Llama-2 tokenizer (Touvron et al., 2023), pretrain on DCLM (Li et al., 2024), and report validation loss on C4-EN (Raffel et al., 2020).

Suite A Results. Figure 2 shows the Suite A pretraining validation-loss curves for Llama-130M and Llama-300M, while Figure 1 reports the corresponding Llama-1.2B result. Under the tuned Suite A protocol (Wen et al., 2026), Muon-NSR and Muon-VS consistently improve over Muon at all three scales. At Llama-130M and Llama-300M, where AdaMuon and NorMuon are also evaluated, both proposed variants achieve lower final validation loss than these adaptive Muon-family baselines, with corresponding gains also visible in the wall-clock curves. At Llama-1.2B, Muon-

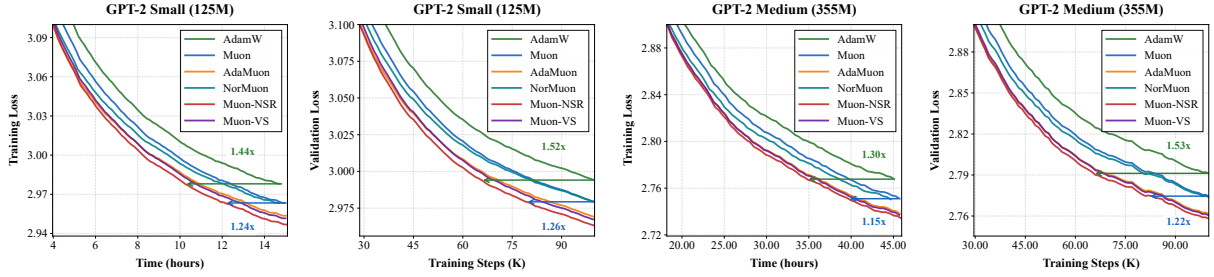


Figure 4: **nanoGPT-based GPT-2 pretraining.** Training loss versus wall-clock time and validation loss versus steps for **GPT-2 Small** and **GPT-2 Medium**, following the AdaMuon setup (Si et al., 2025), with EMA smoothing.

VS outperforms NorMuon and achieves a $1.33\times$ step-to-target speedup over the well-tuned Muon baseline, using the final Muon validation loss as the target. Appendix E and Table 13 further show consistent downstream macro-average gains over the evaluated Muon-family baselines on Suite A Llama-130M and Llama-300M. Appendix B.6 reports an additional comparison with Muon² on Suite A Llama-130M and Llama-300M.

Suite B Setup. Following the Llama pretraining benchmark protocol of Semenov et al. (2025), we evaluate Llama-210M and Llama-720M on the FineWeb sample-100BT subset (Penedo et al., 2024). The learning rate follows a linear-warmup cosine-decay schedule. We compare Muon-NSR and Muon-VS against AdamW and representative Muon-family baselines under the matched benchmark recipe, with AdamW using the tuned benchmark hyperparameters and Muon-family methods using the same hyperparameter settings as Muon.

Suite B Results. Figure 3 reports pretraining validation loss and validation-set next-token top-1 accuracy over training steps for Llama-210M and Llama-720M. Under the matched Suite B protocol, Muon-NSR and Muon-VS consistently rank among the leading methods at both model scales, outperforming AdamW, Muon, AdaMuon, and NorMuon. Using the final Muon validation loss as the target, Muon-NSR achieves step-to-target speedups of $1.09\times$ on Llama-210M and $1.11\times$ on Llama-720M relative to the well-tuned Muon baseline. The improvement is reflected in next-token top-1 accuracy across both settings, suggesting that the gains are not specific to the loss metric but reflect improvements in the optimization trajectory.

5.2 GPT-2 Pretraining

GPT-2 Setup. We pretrain GPT-2 Small (125M) and Medium (355M) on OpenWebText (Gokaslan

and Cohen, 2019) using the AdaMuon (Si et al., 2025) implementation built on nanoGPT (Karpathy, 2022). We compare Muon-NSR and Muon-VS against AdamW and representative Muon-family baselines under the same AdaMuon configuration, with a linear warmup followed by a constant learning rate throughout the remaining training steps.

GPT-2 Results. Figure 4 reports training loss over wall-clock time and validation loss over training steps. Across both GPT-2 Small and GPT-2 Medium, Muon-NSR achieves the fastest training-loss reduction and the lowest final validation loss among the evaluated optimizers. Relative to the Muon baseline, Muon-NSR achieves time-to-target speedups of $1.24\times$ and $1.15\times$ on the training-loss curves and step-to-target speedups of $1.26\times$ and $1.22\times$ on the validation-loss curves for GPT-2 Small and GPT-2 Medium, respectively. For reference, Figure 4 also annotates the AdamW-relative speedups. Muon-VS also attains lower final validation loss than AdamW, Muon, AdaMuon, and NorMuon at both scales. Together, these results show that the benefits of variance-adaptive modulation extend to nanoGPT-based GPT-2 pretraining under the matched AdaMuon setup. Appendix B.6 reports a comparison with Muon² on GPT-2.

5.3 Ablation Studies

Role of the variance coefficient γ . Table 1 examines the variance coefficient γ in Eq. (11). In Muon-NSR, γ weights the variance surrogate $\hat{\Gamma}_t$ in the denominator of Eq. (11), thereby controlling the strength of variance-adaptive damping. We sweep γ on a logarithmic grid while fixing all other settings to assess hyperparameter sensitivity.

The special case $\gamma = 1$ corresponds to the most direct Adam–Muon combination under Adam’s equal-beta variance decomposition, yielding the unit-coefficient form of the pre-orthogonalization normalization in Eq. (11). Table 1 shows that this

Optimizer	Val. Loss ↓	$\Delta_{\text{AdaMuon}} \uparrow$
GPT-2 Small		
AdaMuon (Baseline)	2.9572	–
Muon-NSR ($\gamma = 1$)	2.9531	+0.0041
Muon-NSR ($\gamma = 10$)	2.9508	+0.0064
Muon-NSR ($\gamma = 100$)	2.9536	+0.0036
Muon-NSR ($\gamma = 1000$)	2.9545	+0.0027
Muon-VS	2.9543	+0.0029
Llama-130M		
AdaMuon (Baseline)	3.2126	–
Muon-NSR ($\gamma = 1$)	3.2104	+0.0022
Muon-NSR ($\gamma = 10$)	3.2067	+0.0059
Muon-NSR ($\gamma = 100$)	3.2072	+0.0054
Muon-NSR ($\gamma = 1000$)	3.2067	+0.0059
Muon-NSR ($\gamma = 10000$)	3.2077	+0.0049
Muon-VS	3.2066	+0.0060

Table 1: **Sensitivity to the variance coefficient γ .** Final validation loss and absolute loss reduction Δ_{AdaMuon} are reported relative to AdaMuon in both settings, with Muon-VS representing the large- γ limiting behavior.

direct variant improves upon the AdaMuon baseline in both settings but remains suboptimal. Relaxing this unit coefficient yields lower validation loss on Llama-130M and GPT-2 Small. This gap motivates using γ to control the strength of variance-adaptive damping, rather than fixing it to the value imposed by the combination.

Muon-NSR is robust to the choice of γ . Across GPT-2 Small and Llama-130M, all evaluated values of γ within the tested logarithmic ranges improve over AdaMuon. Although the preferred setting can vary across training recipes, this variation does not indicate a need for fine-grained per-model tuning. The performance variation across the tested values is mild, suggesting that a coarse log-scale choice is sufficient. Thus, γ mainly controls the strength of variance-adaptive damping rather than behaving as a brittle hyperparameter. If eliminating this coefficient is desired, Muon-VS provides a γ -free alternative that removes the additional variance coefficient while retaining variance-based modulation. Additional validation-loss trajectories for representative γ settings are provided in Appendix C.1.

Ordering of NSR modulation. We examine whether variance modulation should be applied before or after Newton–Schulz orthogonalization. Muon-NSR applies the NSR gate in Eq. (11) to the update proxy before the K -step Newton–Schulz map. Under the exact polar map, all nonzero singular values are normalized to one, whereas a nonuniform element-wise gate applied afterward need not preserve this property. Appendix C.2 establishes

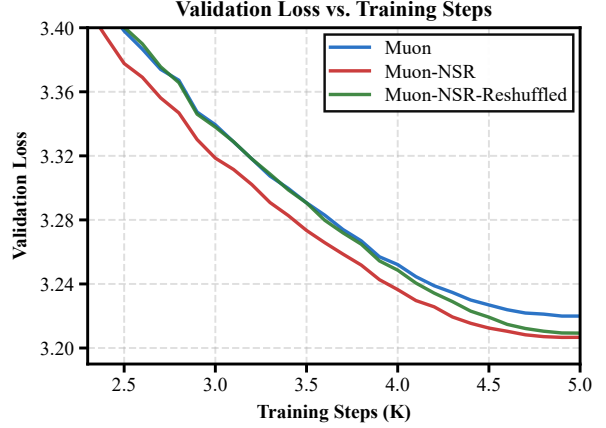


Figure 5: **NSR ordering ablation on Suite A Llama-130M.** Validation loss trajectories of Muon, Muon-NSR, and the reshuffled post-orthogonalization variant.

this structural distinction and derives a finite-step perturbation bound for the Newton–Schulz approximation used in practice. For a controlled comparison, we consider a post-orthogonalization variant that first computes $O_t = \text{NS}_K(\hat{M}_t)$ and then applies the stabilized coordinate-wise NSR gate as follows:

$$S_t = \sqrt{1 + \gamma \frac{\hat{\Gamma}_t}{\hat{M}_t^2 + \varepsilon}}, \quad O_t^{\text{post}} = O_t \oslash S_t,$$

where $\oslash$ denotes element-wise division. Figure 5 shows that applying NSR modulation before orthogonalization yields a smoother optimization trajectory and lower validation loss than applying it afterward on Llama-130M. This comparison empirically supports pre-orthogonalization variance adaptation as the preferred ordering in this setting. Further algorithmic details and GPT-2 Small results are provided in Appendix C.3.

6 Conclusion

We presented Muon-NSR and Muon-VS, demonstrating that Adam-style variance adaptivity can be incorporated into Muon-style matrix updates while preserving Muon’s orthogonalized update geometry. Across Llama-style and GPT-2 pretraining settings at multiple model scales, both variants consistently improve convergence over a well-tuned Muon baseline and representative Muon-family baselines in the evaluated settings, achieving up to a $1.33\times$ step-to-target speedup on Llama-1.2B, with Muon’s final validation loss as the target. These results position variance-adaptive modulation as an optimizer-level mechanism for improving the efficiency of Muon-based language model pretraining.

Limitations

Due to computational constraints, our empirical evaluation covers representative Llama-style and GPT-2 pretraining settings up to 1.2B parameters. These experiments provide controlled, matched comparisons across multiple model scales, but larger-scale settings such as 3B and 7B models remain beyond the scope of this study. Evaluating Muon-NSR and Muon-VS in such regimes would require substantially larger training budgets and carefully matched tuning for all baselines. We leave this larger-scale validation for future work.

Ethical Considerations

This work studies optimizer design for efficient language model pretraining at scale. The study does not involve human-subject data, create new annotations, or collect or release personally identifiable information. We use publicly available pretraining corpora and validation sets in accordance with the corresponding published protocols solely for controlled aggregate model-level pretraining, ablation analysis, and validation-loss evaluation.

The main anticipated benefit is a reduction in the number of optimization steps required to reach a fixed validation-loss target in the evaluated pretraining settings, thereby lowering computational cost under fixed budgets. However, such efficiency gains may also lower the barrier to training large language models, making their broader impact application-dependent. As this study uses publicly available web-scale corpora, it inherits the data-curation limitations of the original datasets.

We use all public datasets and codebases in accordance with their respective licenses and terms of use. We do not redistribute the corpora or any pretrained checkpoints. The released code includes third-party license and attribution information as well as data-preparation instructions for the datasets used in our experiments.

Acknowledgments

H. Li was supported by the NSF China (No. 62476142).

References

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Aziz-zadenesheli, and Animashree Anandkumar. 2018. signSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning (ICML)*.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: Reasoning about physical commonsense in natural language. In *AAAI Conference on Artificial Intelligence*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V Le. 2023. Symbolic discovery of optimization algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

DeepSeek-AI. 2026. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*.

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

Aaron Gokaslan and Vanya Cohen. 2019. OpenWebText corpus. <http://Skylion007.github.io/OpenWebTextCorpus>.

Vineet Gupta, Tomer Koren, and Yoram Singer. 2018. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning (ICML)*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, and 3 others. 2022. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems (NeurIPS)*.

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

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020.

- Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.
- Andrej Karpathy. 2022. nanoGPT. <https://github.com/karpathy/nanoGPT>.
- Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.
- Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Tim Tsz-Kit Lau, Qi Long, and Weijie Su. 2025. PolarGrad: A class of matrix-gradient optimizers from a unifying preconditioning perspective. *arXiv preprint arXiv:2505.21799*.
- Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, and 40 others. 2024. DataComp-LM: In search of the next generation of training sets for language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Zichong Li, Liming Liu, Chen Liang, Weizhu Chen, and Tuo Zhao. 2026. NorMuon: Making Muon more efficient and scalable. In *International Conference on Machine Learning (ICML)*.
- Kaizhao Liang, Lizhang Chen, Bo Liu, and Qiang Liu. 2026. Cautious optimizers: Improving training with one line of code. In *International Conference on Learning Representations (ICLR)*.
- Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
- Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, and 9 others. 2025. Muon is scalable for LLM training. *arXiv preprint arXiv:2502.16982*.
- Ziyue Liu, Ruijie Zhang, Zhengyang Wang, Yequan Zhao, Yupeng Su, Zi Yang, and Zheng Zhang. 2026. Muon²: Boosting Muon via adaptive second-moment preconditioning. *arXiv preprint arXiv:2604.09967*.
- Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*.
- James Martens and Roger Grosse. 2015. Optimizing neural networks with Kronecker-factored approximate curvature. In *International Conference on Machine Learning (ICML)*.
- Antonio Orvieto and Robert Gower. 2025. In search of Adam’s secret sauce. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Guilherme Penedo, Hynek Kydlíček, Loubna Ben al-lal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Boris T Polyak. 1964. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.
- Herbert Robbins and Sutton Monroe. 1951. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. WinoGrande: An adversarial Winograd Schema Challenge at scale. In *AAAI Conference on Artificial Intelligence*.
- Andrei Semenov, Matteo Pagliardini, and Martin Jaggi. 2025. Benchmarking optimizers for large language model pretraining. *arXiv preprint arXiv:2509.01440*.
- Ishaan Shah, Anthony M. Polloreno, Karl Stratos, Philip Monk, Adarsh Chaluvaraju, Andrew Hojel, Andrew Ma, Anil Thomas, Ashish Tanwer, Darsh J. Shah, Khoi Nguyen, Kurt Smith, Michael Callahan, Michael Pust, Mohit Parmar, Peter Rushton, Platon

- Mazarakis, Ritvik Kapila, Saurabh Srivastava, and 4 others. 2025. Practical efficiency of Muon for pre-training. *arXiv preprint arXiv:2505.02222*.
- Chongjie Si, Debing Zhang, and Wei Shen. 2025. AdaMuon: Adaptive Muon optimizer. *arXiv preprint arXiv:2507.11005*.
- Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.
- Lintang Sutawika, Hailey Schoelkopf, Leo Gao, Baber Abbasi, Stella Biderman, Jonathan Tow, ben fattori, Charles Lovering, farzanehnakhaee70, Jason Phang, Anish Thite, Fazz, Thomas Wang, Niklas, Aflah, sdt-blck, nopperl, gakada, tttyuntian, and 11 others. 2026. [EleutherAI/lm-evaluation-harness: v0.4.11](#).
- Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-RMSProp: Divide the gradient by a running average of its recent magnitude. In *COURSERA: Neural Networks for Machine Learning*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. 2025. SOAP: Improving and stabilizing Shampoo using Adam for language modeling. In *International Conference on Learning Representations (ICLR)*.
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Kaiyue Wen, David Hall, Tengyu Ma, and Percy Liang. 2026. Fantastic pretraining optimizers and where to find them. In *International Conference on Learning Representations (ICLR)*.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Conference on Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations*.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
- Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. ReCoRD: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*.

A Experimental Setup Details

Overview and Compute Environment. This section details the Llama-style and GPT-2 pretraining settings considered in this study. We describe the model architectures, data preprocessing, training budgets, batching strategies, optimizer hyperparameters, and evaluation protocols. All pretraining experiments were implemented in PyTorch (Paszke et al., 2019) and conducted on a single node with $8\times$ NVIDIA RTX 5090 GPUs using Distributed Data Parallel (DDP).

A.1 Evaluation Metrics

For each run, we report the final validation loss under the fixed pretraining budget. When reported, validation perplexity is computed as $\text{PPL} = \exp(\mathcal{L}_{\text{val}})$, where $\mathcal{L}_{\text{val}}$ is the mean token-level cross-entropy on the validation set.

We quantify convergence efficiency using step-to-target and time-to-target speedups. For a method m and a baseline b , the speedup at target loss $\ell_\star$ is

$$S_z(m, b; \ell_\star) = \frac{\tau_z(b; \ell_\star)}{\tau_z(m; \ell_\star)}, \quad z \in \{\text{step, time}\},$$

where $\tau_z(a; \ell_\star)$ denotes the first interpolated step or wall-clock time at which method a reaches $\ell_\star$ on the corresponding training- or validation-loss curve. By construction, larger S_z indicates faster convergence. Unless otherwise specified, $\ell_\star$ is the reference baseline’s final loss on the same curve under the same pretraining budget. For the Llama-style multi-target analysis in Appendix B.3, we use Muon’s validation losses at 20% intervals over 20%–100% of its training budget as five shared targets. For readability, the main-text speedups are annotated on the corresponding learning curves.

A.2 Suite A Llama Pretraining Setup

We conduct the Suite A experiments with Llama architectures following the pretraining protocol of Wen et al. (2026), which prioritizes regime-specific hyperparameter optimization for fair benchmarking. The codebase is built on the public pretraining scaffold released with Liang et al. (2026). The model-scale, training-budget, learning-rate-schedule, and Muon hyperparameter settings follow Wen et al. (2026), while the data and evaluation configuration is specified below.

Model Architecture. We pretrain three decoder-only models based on the Llama architecture: Llama-130M, Llama-300M, and Llama-1.2B. To

isolate the impact of width, we keep model depth and context length fixed across all scales. Table 2 provides the detailed architectural specifications.

Data and Preprocessing. We pretrain on DCLM using the Llama-2 tokenizer with sequence packing and token-stream preprocessing. Evaluation is performed on the C4-EN validation split. Within each figure or table, all optimizers use the same preprocessing, training budget, validation split, and training stream to ensure controlled comparisons.

Optimizers and Hyperparameters. We evaluate Muon, NorMuon, Muon-NSR, and Muon-VS at all three Suite A scales, and include AdaMuon for Llama-130M and Llama-300M. Muon-NSR, Muon-VS, and NorMuon reuse the Muon hyperparameters without separate retuning across scales, with $\gamma = 1000$ for Muon-NSR in the main comparisons. For AdaMuon, Appendix B.5 reports a dedicated learning-rate and scaling-rule search on Llama-130M, and the reported AdaMuon results use the selected implementation-level scaling rule $s_{\text{scale}} = \frac{0.2\sqrt{mn}}{\|M_{\text{update}}\|_F}$, where $M_{\text{update}} \in \mathbb{R}^{m \times n}$ denotes AdaMuon’s update matrix. For Muon, Muon-NSR, Muon-VS, and NorMuon, we adopt the same Suite A update-scale rule $s_{\text{scale}} = \sqrt{\max(1, m/n)}$ from Wen et al. (2026).

A.3 Suite B Llama Pretraining Setup

We follow the Suite B Llama pretraining benchmark (Semenov et al., 2025). All models use a decoder-only Llama architecture with Swish-gated linear unit (SwiGLU) feed-forward blocks, root-mean-square layer normalization (RMSNorm; $\varepsilon = 10^{-5}$), Rotary Position Embeddings (RoPE) (Su et al., 2024), tied input-output embeddings, bias-free linear layers, and no dropout. Parameters are initialized from a truncated Gaussian distribution with $\sigma = 0.02$. The model configurations and training hyperparameters are summarized in Table 3.

Data and Preprocessing. We pretrain on the FineWeb sample-100BT subset (Penedo et al., 2024) following the benchmark preprocessing protocol. The corpus is tokenized using the GPT-2 byte-pair encoding (BPE) tokenizer with a vocabulary size of 50,304. Both training and evaluation use a fixed context length of 512 tokens. The Llama-210M and Llama-720M runs use 16.8B and 48.8B training tokens, corresponding to $4\times$ and approximately $3.4\times$ their Chinchilla-optimal budgets of 4.2B and 14.4B tokens, respectively.

Model	Hid	FFN	Heads	BSZ	η_{adam}	η_{muon}	β_{muon}	g_{norm}	(β_1, β_2)	WSD	WU	η_{min}
Llama-130M	512	2048	8	128	0.0032	0.016	0.95	1.0	(0.8, 0.98)	0.8	0	0
Llama-300M	768	3072	12	128	0.0024	0.008	0.98	1.0	(0.8, 0.98)	0.8	0	0
Llama-1.2B	1536	6144	24	256	0.0012	0.008	0.98	2.0	(0.8, 0.98)	1	0	0

Table 2: **Suite A Llama pretraining configuration and optimizer hyperparameters.** All models use 32 Transformer layers and a context length of 4096. Hid denotes the hidden size, FFN denotes the feed-forward network width, Heads denotes the number of attention heads, BSZ denotes the global batch size in sequences, g_{norm} denotes the global gradient-norm threshold, WU denotes warmup steps, and WSD denotes the decay-stage setting in the Suite A protocol’s warmup–stable–decay schedule, which uses linear decay.

Model	Hid	L	H	Batch	Steps	Tok	Sch	WU	Peak	WD	Clip
Llama-210M	768	24	12	256	128k	16.8B	cos	2k	1×10^{-3}	0.1	0.5
Llama-720M	2048	12	16	1984	48k	48.8B	cos	2k	1×10^{-3}	0.1	0.1

Table 3: **Suite B Llama pretraining configuration.** Hid: hidden size; L: layers; H: attention heads; Batch: global batch size in sequences; Tok: training tokens; Sch: learning-rate decay schedule; WU: warmup steps; Peak: peak learning rate; WD: weight decay; Clip: gradient-norm clipping threshold.

Optimization Protocol and Shared Settings.

We compare six optimizers under a matched protocol: AdamW, Muon, AdaMuon, NorMuon, Muon-NSR, and Muon-VS. The model architecture, data, training budget, and learning-rate schedule are held fixed across optimizers within each model configuration. All optimizers use decoupled weight decay with $\lambda_{\text{wd}} = 0.1$ and model-specific global gradient-norm clipping thresholds, as reported in Table 3. The learning rate is linearly warmed up over 2,000 steps to its peak value, followed by cosine decay to 0.01 times the peak learning rate. For Muon-family methods, we inherit the Muon baseline configuration except for method-specific components of the update rule. Following the benchmark implementation (Semenov et al., 2025), matrix-shaped parameters in Transformer blocks are optimized by the corresponding Muon-style optimizer, while embedding tables and vector-shaped parameters are updated by AdamW. The Muon configuration uses momentum $\beta_{\text{muon}} = 0.95$, together with Nesterov acceleration and $K = 5$ Newton–Schulz iterations.

Optimizer-Specific Settings. AdamW uses momentum coefficients $(\beta_1, \beta_2) = (0.9, 0.999)$ for Llama-210M and $(0.9, 0.99)$ for Llama-720M, with $\varepsilon_{\text{adam}} = 10^{-8}$. For Muon, NorMuon, Muon-NSR, and Muon-VS, we use the benchmark Muon update-scaling rule $s_{\text{scale}} = 0.2\sqrt{\max(m, n)}$. AdaMuon instead follows its original RMS-aligned update-scaling rule, $s_{\text{scale}} = \frac{0.2\sqrt{mn}}{\|M_{\text{update}}\|_F}$, as implemented in Si et al. (2025). For Muon-NSR, γ is fixed to 1000 at both model scales.

Evaluation Protocol. We evaluate all models under an identical validation protocol across optimizers. We report validation loss, computed as the mean token-level cross-entropy, and next-token top-1 accuracy, computed as the fraction of token positions where the model’s highest-probability prediction matches the ground-truth token.

A.4 GPT-2 (nanoGPT) Pretraining Setup

We conduct GPT-2 pretraining experiments using the AdaMuon codebase (Si et al., 2025), which builds on the nanoGPT implementation (Karpathy, 2022) of GPT-2 (Radford et al., 2019). Following AdaMuon, we use bias-free linear layers, disable dropout, retain Gaussian Error Linear Unit (GELU) activations, replace learned positional embeddings with RoPE, and replace the standard cosine decay schedule with a 2,000-step linear warmup followed by a constant learning rate. We evaluate GPT-2 Small and GPT-2 Medium, with architectural configurations summarized in Table 4.

Data. We train on OpenWebText (Gokaslan and Cohen, 2019) using the GPT-2 BPE tokenizer (Radford et al., 2019) and evaluate on its validation split.

Training Budget and Batching. All models are trained for 100,000 optimizer updates on 8 GPUs using Distributed Data Parallel (DDP). We use gradient accumulation to obtain an effective batch size of 60 sequences per GPU, corresponding to an effective global batch size of 480 sequences per optimizer update. With a sequence length of 1,024, this training budget contains about 4.92×10^{10} tokens.

Model	Layers	Heads	Hidden size	Parameters
GPT-2 Small	12	12	768	125M
GPT-2 Medium	24	16	1024	355M

Table 4: **GPT-2 model configurations.** Architecture sizes used in the nanoGPT pretraining experiments.

Optimizer	Val. Loss ↓	Val. PPL ↓	$\Delta_{\text{Muon}} \uparrow$
Llama-130M			
Muon	3.2199	25.03	—
AdaMuon	3.2126	24.84	0.0073
NorMuon	3.2115	24.82	0.0084
Muon-NSR	3.2067	24.70	0.0132
Muon-VS	3.2066	24.69	0.0133
Llama-300M			
Muon	3.0177	20.44	—
AdaMuon	3.0109	20.31	0.0068
NorMuon	3.0107	20.30	0.0070
Muon-NSR	3.0046	20.18	0.0131
Muon-VS	3.0011	20.11	0.0166
Llama-1.2B			
Muon	2.7458	15.58	—
NorMuon	2.7367	15.44	0.0091
Muon-NSR	2.7346	15.40	0.0112
Muon-VS	2.7323	15.37	0.0135

Table 5: **Suite A final validation loss and perplexity.** Results are reported for each optimizer after the fixed pretraining budget at each model scale under the Suite A protocol. Perplexity is computed as $\text{PPL} = \exp(\mathcal{L}_{\text{val}})$. $\Delta_{\text{Muon}} = \mathcal{L}_{\text{Muon}} - \mathcal{L}_{\text{opt}}$ denotes the validation-loss reduction relative to Muon at the same model scale. Bold indicates the best value within each scale.

Optimizers and Hyperparameters. We compare our proposed Muon-NSR and Muon-VS with AdamW, Muon, AdaMuon, and NorMuon. All methods use the same learning-rate schedule, with a peak learning rate of 6×10^{-4} , weight decay of 0.1, and a global gradient-norm clipping threshold of 1.0. AdamW uses $(\beta_1, \beta_2) = (0.9, 0.95)$. For Muon-family methods, we follow the hybrid parameter partitioning strategy of Si et al. (2025): matrix-shaped parameters in Transformer blocks are optimized with Muon-style updates, while embedding tables and vector-shaped parameters are updated with AdamW. We set the Muon momentum to 0.95 and use $s_{\text{scale}} = 0.2\sqrt{\max(m, n)}$, where m and n denote the matrix dimensions. AdaMuon and NorMuon instead use the implementation-level scaling factor $s_{\text{scale}} = \frac{0.2\sqrt{mn}}{\|M_{\text{update}}\|_F}$. For Muon-NSR, we set $\gamma = 10$ in the main nanoGPT comparisons; ablation-specific values are specified separately.

Optimizer	Val. Loss ↓	Val. PPL ↓	$\Delta_{\text{AdamW}} \uparrow$
Llama-210M			
AdamW	3.0450	21.01	—
Muon	3.0424	20.96	0.0026
AdaMuon	3.0429	20.97	0.0021
NorMuon	3.0371	20.84	0.0079
Muon-NSR	3.0320	20.74	0.0130
Muon-VS	3.0338	20.78	0.0112
Llama-720M			
AdamW	2.8001	16.45	—
Muon	2.7999	16.44	0.0002
AdaMuon	2.7967	16.39	0.0034
NorMuon	2.7957	16.37	0.0044
Muon-NSR	2.7916	16.31	0.0085
Muon-VS	2.7929	16.33	0.0072

Table 6: **Suite B final validation loss and perplexity.** Results are reported for each optimizer after the fixed pretraining budget at each model scale under the Suite B protocol. Perplexity is computed as $\text{PPL} = \exp(\mathcal{L}_{\text{val}})$. $\Delta_{\text{AdamW}} = \mathcal{L}_{\text{AdamW}} - \mathcal{L}_{\text{opt}}$ denotes the validation-loss reduction relative to AdamW at the same model scale. Bold indicates the best value within each scale.

B Additional Results and Comparisons

B.1 Additional Suite A Final-Loss Results

Table 5 reports the final validation loss and perplexity under the fixed Suite A pretraining budget. Across all three model scales, Muon-NSR and Muon-VS improve over Muon; on Llama-130M and Llama-300M, they also outperform AdaMuon and NorMuon. Together with the step-to-target and wall-clock results in Figures 1, 2, and 6, these results show that pre-orthogonalization variance modulation consistently improves Muon-style optimization under the Suite A protocol.

B.2 Additional Suite B Final-Loss Results

Table 6 reports the Suite B validation loss of AdamW, Muon, AdaMuon, NorMuon, and our proposed variants. Figure 3 shows the corresponding learning curves, including AdaMuon and NorMuon. Across both model scales, Muon yields only marginal gains over AdamW, whereas our variance-adaptive variants deliver clearer and larger reductions. Muon-NSR attains the lowest final loss at both scales, while Muon-VS remains in the leading group and improves over all evaluated baselines. These results indicate that incorporating variance statistics into Muon-style updates lowers final validation loss under a fixed training budget.

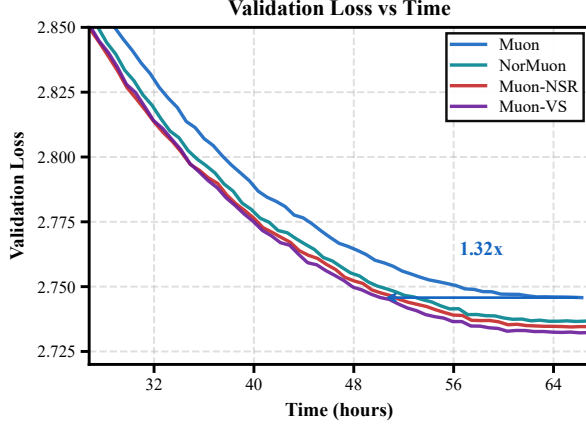


Figure 6: **Suite A Llama-1.2B wall-clock trajectory.** Validation loss on the C4-EN validation split versus wall-clock time during pretraining on DCLM with the $1\times$ Chinchilla token budget.

B.3 Multi-Target Speedups on Llama-Style Pretraining

Because step-to-target and time-to-target speedups depend on the validation-loss target, we complement the single-target speedups reported in the main figures with speedups at five shared validation-loss targets to assess target robustness across all Llama-style settings in Suites A and B. For each setting, we define the five targets as Muon’s validation losses measured at 20%, 40%, 60%, 80%, and 100% of its training budget, and use the same targets to evaluate Muon, Muon-NSR, and Muon-VS. The first step and wall-clock time at which each method reaches a target are estimated by linear interpolation between adjacent evaluations, following Appendix A.1.

Table 8 shows that both proposed variants consistently require fewer training steps and less wall-clock time than Muon to reach every reported target across all evaluated settings in both pretraining suites. However, the magnitude of the speedup varies across target losses and does not necessarily change monotonically as the target is moved later in training. For example, on Suite B Llama-210M, the step-speedup tuple decreases from $(1.14\times, 1.14\times)$ at the 20% target to $(1.09\times, 1.08\times)$ at the 100% target. Taken together with the fixed-budget final-loss results in Tables 5 and 6, the multi-target analysis shows that the observed acceleration is not specific to a single endpoint target. Both variants reach matched validation-loss levels earlier across all five evaluated targets while also attaining lower final validation loss.

Optimizer	Val. Loss ↓	Val. PPL ↓	$\Delta_{\text{Muon}} \uparrow$
GPT-2 Small			
AdamW	2.9810	19.71	-0.0144
Muon	2.9666	19.43	—
AdaMuon	2.9572	19.24	0.0094
NorMuon	2.9671	19.44	-0.0005
Muon-NSR	2.9508	19.12	0.0158
Muon-VS	2.9543	19.19	0.0123
GPT-2 Medium			
AdamW	2.7745	16.03	-0.0187
Muon	2.7558	15.73	—
AdaMuon	2.7429	15.53	0.0129
NorMuon	2.7555	15.73	0.0003
Muon-NSR	2.7404	15.49	0.0154
Muon-VS	2.7418	15.51	0.0140

Table 7: **GPT-2 final validation loss and perplexity.** Results are reported for each optimizer after the fixed 100K-step pretraining budget at each model scale. Perplexity is computed as $\text{PPL} = \exp(\mathcal{L}_{\text{val}})$. $\Delta_{\text{Muon}} = \mathcal{L}_{\text{Muon}} - \mathcal{L}_{\text{opt}}$ denotes the validation-loss reduction relative to Muon at the same model scale. Bold indicates the best value within each scale.

B.4 Additional GPT-2 Final-Loss Results

Table 7 reports the final validation loss and perplexity for GPT-2 Small and GPT-2 Medium under the fixed 100K-step training budget. The values are taken from the final unsmoothed validation log point for each optimizer run, complementing the smoothed training and validation curves in Figure 4. At both model scales, Muon-NSR achieves the lowest validation loss, while Muon-VS also improves over AdamW, Muon, AdaMuon, and NorMuon. These final-loss results confirm that the GPT-2 gains observed in the learning curves persist at the end of training.

B.5 AdaMuon Tuning Details on Suite A

Motivation. The Suite A experiments include AdaMuon as an adaptive Muon-family baseline for Llama-130M and Llama-300M. Directly reusing the Suite A Muon learning rate and update-scale rule is suboptimal for AdaMuon on Llama-130M. We therefore perform a dedicated search over the AdaMuon learning rate and update-scale rule for matrix-shaped parameters on Llama-130M, use the selected configuration at that scale, and transfer the selected scaling rule with a scale-adjusted learning rate to Llama-300M. We also attempted limited AdaMuon trials at Llama-1.2B, but the available compute budget did not permit a broad optimizer-specific search at this larger model scale.

Protocol / Model	Quantity	20%	40%	60%	80%	100%
Suite A Llama-130M	Target loss	3.7320	3.4783	3.3394	3.2521	3.2199
	Step speedup	(1.07, 1.07)	(1.07, 1.07)	(1.05, 1.06)	(1.05, 1.06)	(1.17, 1.16)
	Time speedup	(1.05, 1.05)	(1.06, 1.06)	(1.04, 1.04)	(1.04, 1.04)	(1.15, 1.15)
Suite A Llama-300M	Target loss	3.4120	3.1985	3.0929	3.0334	3.0177
	Step speedup	(1.10, 1.09)	(1.07, 1.08)	(1.08, 1.09)	(1.10, 1.11)	(1.24, 1.28)
	Time speedup	(1.06, 1.05)	(1.03, 1.04)	(1.04, 1.06)	(1.06, 1.07)	(1.20, 1.24)
Suite A Llama-1.2B	Target loss	3.0167	2.8666	2.7901	2.7542	2.7458
	Step speedup	(1.08, 1.07)	(1.08, 1.07)	(1.09, 1.10)	(1.14, 1.15)	(1.30, 1.33)
	Time speedup	(1.11, 1.10)	(1.08, 1.07)	(1.09, 1.10)	(1.13, 1.15)	(1.29, 1.32)
Suite B Llama-210M	Target loss	3.3593	3.2695	3.1849	3.0922	3.0424
	Step speedup	(1.14, 1.14)	(1.10, 1.10)	(1.04, 1.04)	(1.03, 1.03)	(1.09, 1.08)
	Time speedup	(1.14, 1.15)	(1.11, 1.10)	(1.03, 1.03)	(1.03, 1.03)	(1.08, 1.07)
Suite B Llama-720M	Target loss	3.0582	2.9608	2.8896	2.8261	2.7999
	Step speedup	(1.06, 1.06)	(1.07, 1.08)	(1.04, 1.04)	(1.04, 1.04)	(1.11, 1.10)
	Time speedup	(1.10, 1.10)	(1.09, 1.09)	(1.04, 1.05)	(1.04, 1.04)	(1.12, 1.10)

Table 8: **Step-to-target and time-to-target speedups across shared validation-loss targets for Llama-style pretraining.** Targets are selected from Muon’s validation losses at 20% increments from 20% to 100% of its training budget. Each speedup entry is ordered as (Muon-NSR, Muon-VS) relative to Muon (1.00).

Learning Rate	Val. Loss ↓	Val. PPL ↓
Scale I: $s_{\text{scale}} = \frac{0.2\sqrt{mn}}{\ M_{\text{update}}\ _F}$		
1.0×10^{-3}	3.2486	25.75
2.0×10^{-3}	3.2126	24.84
3.0×10^{-3}	3.2143	24.89
4.0×10^{-3}	3.2151	24.91
Scale II: $s_{\text{scale}} = \sqrt{\max(1, m/n)}$		
2.0×10^{-4}	3.2441	25.64
3.0×10^{-4}	3.2266	25.19
4.0×10^{-4}	3.2230	25.10
5.0×10^{-4}	3.2258	25.17

Table 9: **AdaMuon learning-rate and scaling-rule search on Suite A Llama-130M.** Validation loss and perplexity are reported for each configuration. Perplexity is $\text{PPL} = \exp(\mathcal{L}_{\text{val}})$. The best configuration used in the Suite A comparison is highlighted in bold.

AdaMuon search protocol. Table 9 reports two learning-rate sweeps under two update-scale rules: the AdaMuon implementation-level rule, $s_{\text{scale}} = \frac{0.2\sqrt{mn}}{\|M_{\text{update}}\|_F}$, and the Suite A Muon rule, $s_{\text{scale}} = \sqrt{\max(1, m/n)}$. Because these rules induce different update magnitudes, we use rule-specific learning-rate ranges, with all other settings fixed under the Suite A protocol. The best overall configuration uses the AdaMuon implementation-level scaling rule with a learning rate of 2.0×10^{-3} , which we adopt for Llama-130M. For Llama-300M, we retain the same scaling rule and set the AdaMuon learning rate to 1.3×10^{-3} based on the Llama-130M tuning result and model-scale extrapolation.

Optimizer	Main Run ↓	Repeated Run ↓	Δ_{repeat}
Suite B Llama-210M			
Muon-NSR	3.0320	3.0328	0.0008
Muon-VS	3.0338	3.0328	0.0010
Suite B Llama-720M			
Muon-NSR	2.7916	2.7945	0.0029
Muon-VS	2.7929	2.7927	0.0002
GPT-2 Small			
Muon-NSR	2.9508	2.9502	0.0006
Muon-VS	2.9543	2.9550	0.0007
GPT-2 Medium			
Muon-NSR	2.7404	2.7393	0.0011
Muon-VS	2.7418	2.7417	0.0001

Table 10: **Repeat-run consistency check.** We repeat Muon-NSR and Muon-VS on Suite B and GPT-2 using the main-experiment recipe, hyperparameters, and nominal random seed. We report main- and repeat-run validation losses; Δ_{repeat} is their absolute difference.

B.6 Supplementary Comparison with Muon² on GPT-2 and Suite A

Motivation. Muon² is a recent Muon-family optimizer that preconditions the momentum proxy with an Adam-style uncentered second-moment estimate before Newton–Schulz orthogonalization, making it a relevant pre-orthogonalization adaptive baseline. By contrast, Muon² targets better conditioning of this input, whereas Muon-NSR and Muon-VS are motivated by Adam’s variance-adaptive sign-update interpretation and use on-line centered variance estimates to attenuate noise-dominated entries before orthogonalization.

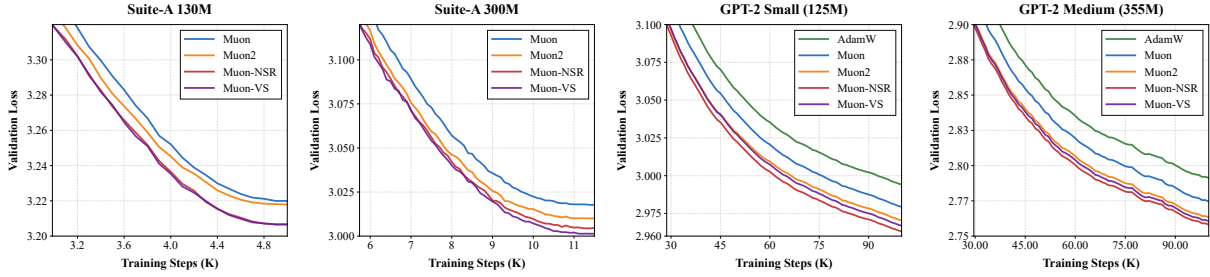


Figure 7: **Supplementary Muon² comparison on Suite A and GPT-2.** Validation loss over training steps for Suite A Llama-130M, Suite A Llama-300M, GPT-2 Small (125M), and GPT-2 Medium (355M). All panels compare Muon, Muon², Muon-NSR, and Muon-VS; the GPT-2 panels additionally include AdamW. Lower is better.

Setup. We compare validation loss over training steps across four representative settings: Suite A Llama-130M, Suite A Llama-300M, GPT-2 Small, and GPT-2 Medium. For each setting, Muon² follows the same model, data, training budget, learning-rate schedule, and parameter-partitioning scheme as the corresponding Muon-family setting. This comparison is intended as a matched-protocol evaluation rather than a fully retuned optimizer study. We also evaluated Muon² on Suite A Llama-1.2B using the corresponding Muon configuration. Because its trajectory closely overlaps with Muon and no optimizer-specific tuning was conducted at this scale, we omit this curve from the figure.

Findings. Figure 7 shows that Muon² improves over Muon across all four supplementary settings, confirming it as a competitive adaptive baseline before orthogonalization. Under the same matched protocols, Muon-NSR and Muon-VS achieve lower final validation loss than Muon² across all four settings. This trend is observed across both the Suite A Llama and GPT-2 experiments, despite their different model families and training recipes. This supplementary comparison therefore extends the matched-protocol evaluation to a recent adaptive Muon-family baseline that also operates before orthogonalization and provides additional evidence for the effectiveness of the proposed variants under matched training conditions.

B.7 Repeat-Run Consistency and Multi-Seed Robustness

Same-nominal-seed repeat runs. Because repeating all optimizer-scale combinations across multiple independent seeds is computationally impractical, we use same-nominal-seed reruns as a repeatability check rather than an independent-seed robustness analysis. We perform one additional end-to-end run for each of Muon-NSR and Muon-

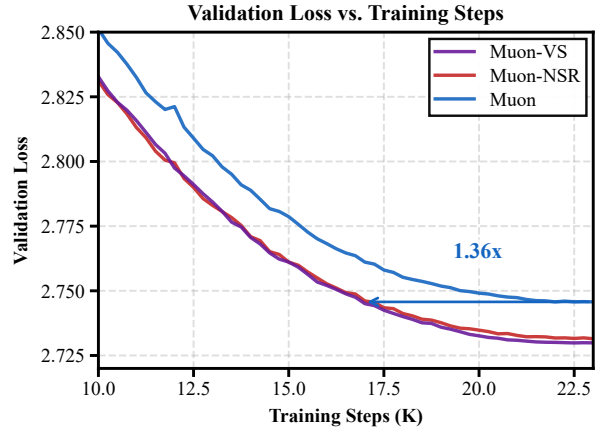


Figure 8: **Data-subset robustness on Suite A Llama-1.2B.** Validation loss over training steps on a different fixed DCLM subset, with all other settings matched to the primary Suite A Llama-1.2B comparison.

VS on four settings with relatively small performance margins: Suite B Llama-210M, Suite B Llama-720M, GPT-2 Small, and GPT-2 Medium. Each repeated run uses the same data, model configuration, training budget, learning-rate schedule, parameter partitioning, optimizer hyperparameters, and nominal seed as the corresponding main experiment. As shown in Table 10, all repeated runs remain close to their main-run counterparts, with $\Delta_{\text{repeat}} \leq 0.0029$. They also remain below the corresponding non-proposed baselines reported in Tables 6 and 7. These small deviations support the repeatability of the reported results under the same nominal seed and do not alter the conclusions.

Data-subset robustness on Suite A Llama-1.2B.

Because the DCLM corpus substantially exceeds the $1\times$ Chinchilla training budget used in Suite A, we additionally assess sensitivity to the sampled training data by evaluating Muon, Muon-NSR, and Muon-VS on an alternative fixed DCLM subset, with all other experimental settings held fixed. As

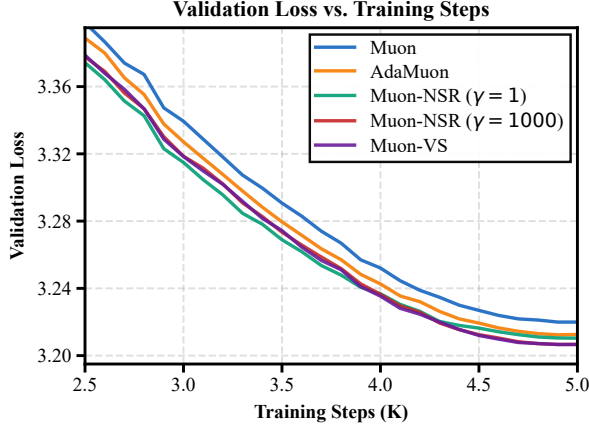


Figure 9: **Representative γ trajectories on Suite A Llama-130M.** Validation loss for Muon, AdaMuon, Muon-NSR ($\gamma \in \{1, 1000\}$), and Muon-VS.

shown in Figure 8, Muon-VS achieves a $1.36\times$ step-to-target speedup over Muon on this alternative subset, compared with $1.33\times$ in the primary four-optimizer comparison in Figure 1. The comparable gains across the two fixed subsets suggest that the observed acceleration is not driven by a particular sampled DCLM subset.

Eight-seed robustness on Suite A Llama-130M. For computational tractability, we conduct the eight-seed evaluation on Suite A Llama-130M, comparing Muon, AdaMuon, NorMuon, Muon², Muon-NSR, and Muon-VS under a matched protocol. All optimizers use the same eight random seeds, with the model, data, training budget, scheduler, parameter partitioning, and evaluation settings matched across optimizers and optimizer-specific hyperparameters fixed across seeds. At each checkpoint, we report mean validation loss with its point-wise two-sided 95% Student- t confidence interval, $\bar{L} \pm t_{0.975, 7} \frac{s}{\sqrt{8}}$, where s is the sample standard deviation across seeds. As shown in Table 11, Muon-NSR and Muon-VS achieve the two lowest mean validation losses at all five reported checkpoints (1K–5K steps), supporting the robustness of the improvements across random seeds.

C Analysis and Ablation Studies

C.1 Validation-Loss Trajectories for γ

To complement the final-loss comparison in Table 1, Figure 9 shows validation-loss trajectories for representative γ settings on Suite A Llama-130M. The direct Adam–Muon variant, Muon-NSR ($\gamma = 1$), achieves lower final validation loss than both Muon and AdaMuon, but remains above

Algorithm 2 Muon-NSR-Reshuffled.

Require: Initial parameter $W_0 \in \mathbb{R}^{m \times n}$; learning-rate schedule $\{\eta_t\}_{t \geq 1}$; weight decay λ_{wd} ; shared EMA rate β ; NSR coefficient γ ; stabilizer ε ; update scale s_{scale} ; Newton–Schulz operator $\text{NS}_K(\cdot)$.

- 1: $M_0 \leftarrow 0, \quad \Gamma_0 \leftarrow 0$
- 2: **for** $t = 1, 2, \dots$ **do**
- 3: $G_t \leftarrow \nabla_W \mathcal{L}(W_{t-1})$
- 4: $M_t \leftarrow \beta M_{t-1} + (1 - \beta)G_t$
- 5: $\Gamma_t \leftarrow \beta \Gamma_{t-1} + \beta(1 - \beta)(M_{t-1} - G_t)^2$
- 6: $\widehat{M}_t \leftarrow M_t / (1 - \beta^t), \quad \widehat{\Gamma}_t \leftarrow \Gamma_t / (1 - \beta^t)$
- 7: $\widetilde{M}_t \leftarrow G_t + \frac{\beta}{1 - \beta} \widehat{M}_t$
- 8: $O_t \leftarrow \text{NS}_K(\widetilde{M}_t)$
- 9: $S_t \leftarrow \sqrt{1 + \gamma \frac{\widehat{\Gamma}_t}{\widetilde{M}_t^2 + \varepsilon}} \text{ {element-wise}}$
- 10: $O_t^{\text{post}} \leftarrow O_t \oslash S_t \text{ {element-wise}}$
- 11: $W_t \leftarrow W_{t-1}(1 - \eta_t \lambda_{\text{wd}}) - \eta_t s_{\text{scale}} O_t^{\text{post}}$
- 12: **end for**

Muon-NSR ($\gamma = 1000$) and Muon-VS. This result suggests that direct Adam-style variance normalization is beneficial, but does not recover the best performance observed among the evaluated variance-adaptive settings. These trajectories are consistent with the sensitivity results in Table 1.

C.2 Pre- and Post-Orthogonalization Variance Modulation

We formalize the distinction between pre- and post-orthogonalization variance modulation in terms of singular-value structure. Applying an element-wise gate before the exact polar map yields a polar update with equal nonzero singular values, whereas a nonuniform gate applied afterward need not preserve this property. We derive a perturbation bound relating the Newton–Schulz approximation error to deviations from this singular-value structure in the practical finite-step setting.

Let $Z \in \mathbb{R}^{m \times n}$ have rank $r \geq 1$, with compact singular value decomposition

$$Z = U_Z \Sigma_Z V_Z^\top.$$

The canonical partial-isometry polar map is defined by

$$\mathcal{P}(Z) = U_Z V_Z^\top = U_Z I_r V_Z^\top.$$

Hence, all nonzero singular values of $\mathcal{P}(Z)$ are equal to one:

$$\sigma_i(\mathcal{P}(Z)) = 1, \quad i = 1, \dots, r.$$

Optimizer	20% (1K)	40% (2K)	60% (3K)	80% (4K)	100% (5K)
Muon	3.7292 $\pm$ 0.0098	3.4733 $\pm$ 0.0049	3.3376 $\pm$ 0.0022	3.2488 $\pm$ 0.0026	3.2182 $\pm$ 0.0028
AdaMuon	3.7418 $\pm$ 0.0047	3.4662 $\pm$ 0.0026	3.3276 $\pm$ 0.0020	3.2405 $\pm$ 0.0021	3.2115 $\pm$ 0.0020
NorMuon	3.7090 $\pm$ 0.0062	3.4603 $\pm$ 0.0024	3.3271 $\pm$ 0.0027	3.2395 $\pm$ 0.0031	3.2095 $\pm$ 0.0035
Muon ²	3.7119 $\pm$ 0.0075	3.4573 $\pm$ 0.0044	3.3234 $\pm$ 0.0029	3.2393 $\pm$ 0.0039	3.2126 $\pm$ 0.0040
Muon-NSR	3.7015 $\pm$ 0.0060	3.4498 $\pm$ 0.0031	3.3177 $\pm$ 0.0027	3.2324 $\pm$ 0.0031	3.2036 $\pm$ 0.0024
Muon-VS	3.7001 $\pm$ 0.0070	3.4517 $\pm$ 0.0027	3.3194 $\pm$ 0.0037	3.2330 $\pm$ 0.0034	3.2037 $\pm$ 0.0032

Table 11: **Eight-seed validation-loss results on Suite A Llama-130M.** Entries report mean validation loss over the same eight random seeds across optimizers $\pm$ the half-width of the pointwise two-sided 95% Student- t confidence interval. Bold marks the lowest mean validation loss at each checkpoint.

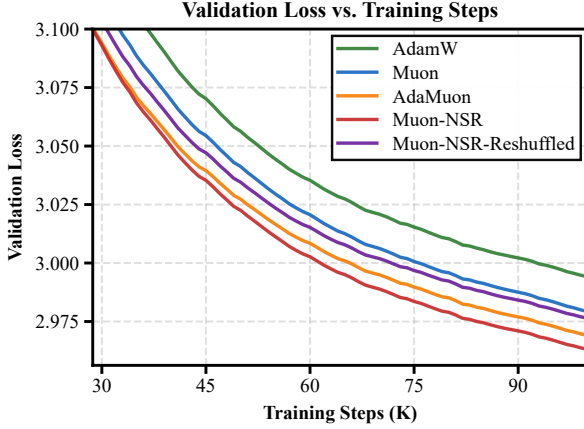


Figure 10: **NSR ordering ablation on GPT-2 Small.** Validation-loss trajectories for Muon-NSR and its post-orthogonalization variant, Muon-NSR-Reshuffled, with AdamW, Muon, and AdaMuon as baselines.

Modulation before the polar map. Let $D \in \mathbb{R}_{>0}^{m \times n}$ be a positive element-wise gate, and define the modulated proxy

$$Y = D \odot Z.$$

Let $q = \text{rank}(Y)$ and write its compact SVD as

$$Y = U_Y \Sigma_Y V_Y^\top.$$

The polar map applied after modulation gives

$$\mathcal{P}(D \odot Z) = \mathcal{P}(Y) = U_Y V_Y^\top = U_Y I_q V_Y^\top.$$

Therefore,

$$\sigma_i(\mathcal{P}(D \odot Z)) = 1, \quad i = 1, \dots, q.$$

Thus, although the element-wise gate may alter the rank and singular structure of the proxy, the subsequent polar map normalizes all nonzero singular values to one. Applying a common update scale $s_{\text{scale}} > 0$ then gives

$$\sigma_i(s_{\text{scale}} \mathcal{P}(D \odot Z)) = s_{\text{scale}}, \quad i = 1, \dots, q,$$

so all nonzero singular values remain equal.

Modulation after the polar map. The equal-nonnzero-singular-value property is not generally preserved when a nonuniform element-wise gate is applied after the polar map. Consider the square case $Z = I_n$ with $n \geq 2$, for which

$$\mathcal{P}(I_n) = I_n.$$

Let $D \in \mathbb{R}_{>0}^{n \times n}$ have unequal diagonal entries $d_1, \dots, d_n$. Then

$$D \odot \mathcal{P}(Z) = D \odot I_n = \text{diag}(d_1, \dots, d_n).$$

Its singular values are $d_1, \dots, d_n$, up to ordering, and hence are not all equal. Thus, nonuniform modulation after orthogonalization need not preserve equality among the nonzero singular values.

Finite-step Newton–Schulz approximation. In practice, Muon uses the K -step Newton–Schulz map NS_K rather than the exact polar map. For the modulated proxy $Y = D \odot Z$ with rank q , define the approximation error

$$\delta_K(Y) = \|\text{NS}_K(Y) - \mathcal{P}(Y)\|_2.$$

The Weyl singular-value perturbation inequality provides a uniform bound for all $i = 1, \dots, q$:

$$|\sigma_i(\text{NS}_K(Y)) - \sigma_i(\mathcal{P}(Y))| \leq \delta_K(Y).$$

Since every nonzero singular value of $\mathcal{P}(Y)$ equals one, the preceding inequality directly yields

$$|\sigma_i(\text{NS}_K(Y)) - 1| \leq \delta_K(Y), \quad i = 1, \dots, q.$$

For each $i = 1, \dots, q$, scaling by $s_{\text{scale}} > 0$ gives

$$|\sigma_i(s_{\text{scale}} \text{NS}_K(Y)) - s_{\text{scale}}| \leq s_{\text{scale}} \delta_K(Y).$$

Thus, operator-norm proximity to the exact polar factor bounds deviations of the nonzero singular values from the equal-singular-value profile.

Muon-NSR and Muon-VS modulate the update proxy with positive element-wise variance gates before Newton–Schulz orthogonalization. Under the exact polar map, this ordering preserves the equal-nonzero-singular-value structure of the update; with finite-step Newton–Schulz, the perturbation bound above controls deviations from this idealized structure. By contrast, Muon-NSR-Reshuffled applies a generally nonuniform element-wise gate after orthogonalization. The preceding counterexample shows that even exact orthogonalization does not ensure preservation of the equal-nonzero-singular-value structure after such post-orthogonalization modulation. This establishes a structural asymmetry between the two orderings and provides a geometric rationale for applying variance modulation before orthogonalization.

C.3 Muon-NSR-Reshuffled

For completeness, Algorithm 2 specifies Muon-NSR-Reshuffled, the post-orthogonalization variant used in the controlled ordering ablation. Compared with Muon-NSR, this variant applies a corresponding coordinate-wise NSR attenuation at the post-orthogonalization stage, enabling a direct controlled assessment of the effect of modulation ordering while keeping all remaining optimization components fixed.

GPT-2 Small ordering ablation. To examine whether the benefit of pre-orthogonalization NSR modulation extends beyond Llama-style pretraining, we evaluate Muon-NSR-Reshuffled on GPT-2 Small (125M) under the nanoGPT-based setup described in Appendix A.4. As shown in Figure 10, the reshuffled variant achieves lower validation loss than AdamW and Muon, showing that post-orthogonalization NSR attenuation can still improve over these baselines. However, it consistently underperforms Muon-NSR and remains above AdaMuon later in training. Together with the Llama-130M ablation, this result provides empirical support for the structural argument in Appendix C.2 and favors applying NSR modulation before Newton–Schulz orthogonalization.

D Runtime and Memory Overhead

We quantify the runtime and memory overhead introduced by the additional variance buffer and element-wise modulation in Muon-NSR and Muon-VS on a single node with $8 \times$ NVIDIA RTX 5090 GPUs. We report padded-token throughput, av-

erage wall-clock time per optimizer update, peak GPU memory, and optimizer-state memory. Measurements for Suite A, Suite B, and GPT-2 use their corresponding training configurations. For Suite A and GPT-2, each microbenchmark averages 200 measured optimizer updates following 10 warmup updates. For Suite B Llama-210M and Llama-720M, we report the average wall-clock time per optimizer update from dedicated 200-update measurement runs. Token throughput is computed using padded tokens under fixed-length training.

Token throughput for optimizer o is computed as

$$\text{Tokens/s}(o) = \frac{B_{\text{eff}} \cdot L}{\bar{t}_{\text{update}}(o)},$$

where B_{eff} is the effective global batch size in sequences per optimizer update, L is the sequence length, and $\bar{t}_{\text{update}}(o)$ is the average wall-clock time per optimizer update. For Suite A and GPT-2 microbenchmarks, $B_{\text{eff}} = B_{\text{micro}} N_{\text{gpu}} A_{\text{grad}}$ under the corresponding training configuration; for Suite B measurements, B_{eff} is the global batch size of the corresponding 200-update run. Optimizer-state memory is reported as the maximum per-rank storage of optimizer-state tensors:

$$M_{\text{opt}} = \frac{1}{2^{30}} \max_r \sum_{s \in \mathcal{S}_{\text{opt}}^{(r)}} \text{numel}(s) \cdot \text{bytes}(s),$$

where $\mathcal{S}_{\text{opt}}^{(r)}$ is the set of optimizer-state tensors stored on rank r , and $\text{bytes}(s)$ denotes the number of bytes per element of tensor s .

As shown in Table 12, these single-node measurements characterize implementation overhead. Across the Suite A, Suite B, and GPT-2 settings considered in this evaluation, Muon-NSR and Muon-VS exhibit only modest changes in per-update wall-clock time while consistently maintaining token throughput close to Muon. This empirical behavior is consistent with the complexity analysis in Section 4, where the additional $O(mn)$ element-wise variance modulation is dominated by the $O(Kmn \min(m, n))$ Newton–Schulz computation for typical matrix-shaped Transformer parameter groups.

The additional variance buffer increases the optimizer-state memory and moderately increases peak GPU memory in these measurements. The variance buffer is maintained only for matrix-shaped parameters in Transformer blocks using Muon updates, while embedding tables and vector-

Model	Optimizer	Tokens/s (K) $\uparrow$	ms/update $\downarrow$	Peak GPU Mem. (GiB) $\downarrow$	Opt. State Mem. (GiB) $\downarrow$
Suite A 130M	Muon	458.3	1144.00	15.41	0.37
	Muon-NSR	447.6	1171.41	15.66	0.62
	Muon-VS	450.4	1164.06	15.66	0.62
Suite A 300M	Muon	274.1	1912.60	17.55	0.75
	Muon-NSR	270.8	1936.07	18.11	1.31
	Muon-VS	271.6	1930.59	18.11	1.31
Suite A 1.2B	Muon	99.5	10535.07	19.28	2.62
	Muon-NSR	99.6	10526.42	21.53	4.87
	Muon-VS	99.7	10520.68	21.53	4.87
Suite B 210M	Muon	416.2	314.89	15.37	0.92
	Muon-NSR	400.6	327.22	16.01	1.55
	Muon-VS	408.2	321.08	16.03	1.55
Suite B 720M	Muon	207.3	4899.25	19.72	3.06
	Muon-NSR	205.7	4939.20	22.02	5.36
	Muon-VS	206.1	4929.15	22.01	5.36
GPT-2 Small	Muon	935.0	525.70	20.51	0.69
	Muon-NSR	927.0	530.24	20.55	0.73
	Muon-VS	925.9	530.84	20.55	0.73
GPT-2 Medium	Muon	305.1	1611.23	16.99	1.03
	Muon-NSR	304.4	1614.65	17.13	1.17
	Muon-VS	304.4	1614.46	17.13	1.17

Table 12: **Runtime and memory overhead on Suite A, Suite B, and GPT-2 models.** We report padded-token throughput, average wall-clock time per optimizer update, peak GPU memory, and maximum per-rank optimizer-state memory. Suite A and GPT-2 rows are measured by microbenchmarks over 200 updates after 10 warmup updates; Suite B rows use dedicated 200-update runs. All measurements use the effective global batch size for each training configuration. Tokens/s is in thousands, ms/update in milliseconds, and memory in GiB.

shaped parameters remain under AdamW and receive no additional variance state. Thus, this additional state is confined exclusively to the Muon-optimized parameter groups. The higher observed peak memory of GPT-2 Small relative to GPT-2 Medium stems from its larger per-GPU microbatch under a different gradient accumulation setting, which increases activation memory despite the smaller model.

E Small-Model Zero-Shot Downstream Evaluation

To assess whether pretraining gains transfer to zero-shot downstream performance, we evaluate the Suite A Llama-130M and Llama-300M checkpoints using EleutherAI’s Language Model Evaluation Harness (lm-eval-harness) (Sutawika et al., 2026). We use lm-eval-harness v0.4.11 with Hugging Face model loading (Wolf et al., 2020), bfloat16 inference, automatic batch-size selection, and the bundled task configurations, including their default prompts, splits, post-processing rules, and metrics. All evaluations are conducted in the zero-shot setting with num_fewshot=0.

Given the relatively small model scales, we treat the downstream evaluation as a targeted sanity check rather than a comprehensive assessment of model capabilities. Accordingly, we report results on a ten-task subset spanning commonsense reasoning, language modeling, reading comprehension, natural language inference, and truthfulness-oriented multiple-choice tasks.

For reproducibility, we report the task metadata versions used in our lm-eval-harness evaluation: v1.0 for arc_easy, hellaswag, lambada_openai, piqa, and winogrande; v2.0 for race, record, and truthfulqa_mc1; v3.0 for truthfulqa_mc2; and v0.0 for sglue_rte.

Table 13 summarizes the small-model zero-shot transfer results. On Llama-130M, Muon-NSR and Muon-VS raise Muon’s unweighted macro-average of 39.35 to 40.83 and 40.65, gains of +1.48 and +1.30 percentage points, respectively. On Llama-300M, they raise it from 41.94 to 42.36 and 42.28, gains of +0.42 and +0.34 percentage points, respectively. At both scales, the proposed variants outperform all evaluated Muon-family baselines in the unweighted macro-average across tasks.

Llama-130M						
Task	Muon	AdaMuon	NorMuon	Muon ²	Muon-NSR	Muon-VS
ARC-E (Clark et al., 2018)	36.95	38.38	37.50	37.04	38.38	37.63
HellaSwag (Zellers et al., 2019)	28.08	28.39	28.49	28.08	28.93	28.60
LAMBADA (Paperno et al., 2016)	24.86	25.91	25.75	25.15	26.12	26.28
PIQA (Bisk et al., 2020)	59.85	60.39	59.85	60.50	59.63	59.19
RACE (Lai et al., 2017)	25.65	26.32	26.41	26.03	27.75	26.89
ReCoRD (Zhang et al., 2018)	52.90	52.72	53.90	53.46	54.57	53.89
RTE (Wang et al., 2019)	50.18	50.54	49.82	52.71	53.79	55.23
TruthfulQA-MC1 (Lin et al., 2022)	22.77	23.99	23.62	24.24	24.24	24.11
TruthfulQA-MC2 (Lin et al., 2022)	40.99	43.52	42.36	42.94	43.36	43.17
WinoGrande (Sakaguchi et al., 2020)	51.30	51.07	51.78	50.91	51.54	51.46
Average	39.35	40.12	39.95	40.11	40.83	40.65

Llama-300M						
Task	Muon	AdaMuon	NorMuon	Muon ²	Muon-NSR	Muon-VS
ARC-E (Clark et al., 2018)	41.16	41.50	40.70	41.08	41.08	40.49
HellaSwag (Zellers et al., 2019)	31.11	31.40	30.99	31.45	31.61	31.25
LAMBADA (Paperno et al., 2016)	30.35	30.23	30.55	30.39	31.13	30.68
PIQA (Bisk et al., 2020)	61.86	62.40	61.64	62.30	61.70	62.46
RACE (Lai et al., 2017)	27.66	26.99	28.61	26.51	28.13	27.37
ReCoRD (Zhang et al., 2018)	58.57	59.11	59.23	58.25	59.53	59.81
RTE (Wang et al., 2019)	52.71	53.07	51.99	51.62	54.87	53.79
TruthfulQA-MC1 (Lin et al., 2022)	22.40	23.01	22.15	21.79	22.64	22.77
TruthfulQA-MC2 (Lin et al., 2022)	41.60	42.98	41.59	40.54	42.82	42.51
WinoGrande (Sakaguchi et al., 2020)	51.93	51.38	50.83	49.88	50.12	51.62
Average	41.94	42.21	41.83	41.38	42.36	42.28

Table 13: **Small-model zero-shot transfer on Suite A Llama checkpoints.** Results are reported as percentages on the ten-task lm-eval-harness subset. ARC-E, HellaSwag, and PIQA use acc_{norm} , ReCoRD uses F_1 , and all other tasks use acc . Average denotes the unweighted macro-average across the ten tasks. AdaMuon, NorMuon, and Muon² serve as Muon-family baselines at both model scales. Boldface in the Average row indicates the two highest macro-average scores at each model scale. Higher values are better.

Task-level gains are mixed, with several evaluated commonsense and physical-reasoning tasks remaining close to or slightly below the Muon baseline for one or both proposed variants. Nevertheless, both Muon-NSR and Muon-VS improve the unweighted macro-average relative to all evaluated Muon-family baselines at both model scales. Taken together with the pretraining results, these findings suggest that variance-adaptive Muon updates improve validation loss and convergence efficiency while preserving and, on average, improving zero-shot transfer in this small-model setting.