Nora: Normalized Orthogonal Row Alignment for Scalable Matrix Optimizer

Jinghui Yuan[1]*, **Jiaxuan Zou**[2,4]*, **Shuo Wang**[3]*, **Yong Liu**[4]†, **Feiping Nie**[1]
1. School of Artificial Intelligence, Optics and Electronics (iOPEN),
Northwestern Polytechnical University.
2. School of Mathematics and Statistics, Xi'an Jiaotong University.
3. Institute for Interdisciplinary Information Sciences, Tsinghua University.
4. Gaoling School of Artificial Intelligence, Renmin University of China.
yuanjh@mail.nwpu.edu.cn, jiaxuanzou@stu.xjtu.edu.cn, runner21st@gmail.com,
liuyonggsai@ruc.edu.cn, feipingnie@gmail.com

Abstract

Matrix-based optimizers have demonstrated immense potential in training Large Language Models (LLMs), however, designing an ideal optimizer remains a formidable challenge. A superior optimizer must satisfy three core desiderata: efficiency, achieving Muon-like preconditioning to accelerate optimization; stability, strictly adhering to the scale-invariance inherent in neural networks; and speed, minimizing computational overhead. While existing methods address these aspects to varying degrees, they often fail to unify them—either incurring prohibitive computational costs like Muon, or allowing radial jitters that compromise stability like RMNP. To bridge this gap, we propose Nora, an optimizer that rigorously satisfies all three requirements. Nora achieves training stability by explicitly stabilizing weight norms and angular velocities through row-wise momentum projection onto the orthogonal complement of the weights. Simultaneously, by leveraging the block-diagonal dominance of the Transformer Hessian, Nora effectively approximates structured preconditioning while maintaining an optimal computational complexity of $\mathcal{O}(mn)$. Furthermore, we prove that Nora is a scalable optimizer and establish its corresponding scaling theorems. With a streamlined implementation requiring only two lines of code, our preliminary experiments validate Nora as an efficient and highly promising optimizer for large-scale training.

1 Introduction

The training of Large Language Models (LLMs) relies heavily on adaptive optimizers such as Adam [1, 2]. However, these methods primarily estimate diagonal curvature, neglecting the rich off-diagonal structural information inherent in the loss landscape [3]. Recently, matrix-based optimizers like Muon have emerged to bridge this gap, achieving state-of-the-art data efficiency through orthogonalized updates [4]. Despite its remarkable performance, Muon relies on the Newton-Schulz iteration [5], which introduces significant computational overhead of $\mathcal{O}(m^2n)$, where the weight matrix $w \in \mathbb{R}^{m\times n}$ typically satisfies $m < n$ in large-scale settings.

Subsequent research has sought to approximate this preprocessor. [6] identified the row-wise block-diagonal dominance of the Transformer Hessian, a phenomenon also noted by [7], who proposed a simplified preprocessor approximating Muon via diagonal matrices. This led to the development of RMNP, a Row-wise Momentum Normalization method. Although RMNP is computationally superior,

*These authors contributed equally.
†Corresponding Author.

Preprint.

it overlooks a fundamental property of modern neural networks: weight space symmetry [8]. Due to the pervasive use of BatchNorm [9], RMSNorm [10] and LayerNorm [11], network representations exhibit scale invariance [12]. Radial updates—those aligning with the weight vector—do not alter the functional output [13]. However, radial momentum noise can interfere with the preprocessor's output and the direction of weight learning. Simultaneously, it silently perturbs the weight norm, leading to chaotic oscillations in the effective learning rate and ultimately undermining network training [14].

Beyond these, other classical or state-of-the-art optimizers typically suffer from one or more deficiencies: they either neglect the use of preconditioners [15, 16, 17], fail to account for the scale-invariance of neural networks [18, 19], or incur excessive computational complexity [20, 21]. In fact, many existing methods simultaneously lack several of these essential properties.

Our motivation stems from a commitment to the three core principles of optimizer design: efficiency, stability, and speed. To this end, we introduce Nora. The design of Nora relies on a simple row-wise orthogonality property: row-wise normalization preserves row-wise perpendicularity to the weight. This property enables us to unify preconditioning estimation, scale-invariance, and rapid computation. Specifically, we first project the momentum v_t onto its component that is row-wise perpendicular to w_t to obtain $v_t^{r\perp}$, which ensures training stability through orthogonality. Subsequently, we apply a diagonal preconditioning estimation to $v_t^{r\perp}$ to derive the update d_t. This step efficiently approximates the Muon-like preconditioning for enhanced efficiency, yet computationally simplifies to a mere row-wise normalization, thereby simultaneously satisfying both the speed and stability requirements.

Furthermore, leveraging the Maximal Update Parametrization (μP) framework [22], we demonstrate that Nora is a scalable optimizer and derive its optimal learning rate scaling laws. We also provide a rigorous convergence analysis, establishing theoretical guarantees for Nora in non-convex optimization landscapes. Remarkably, the core logic of Nora can be implemented in just two lines of matrix-based code, ensuring its seamless compatibility as a plug-and-play module for any existing optimizer. We evaluate Nora by training LLaMA models [23] of various scales across a range of learning rates. Our experimental results demonstrate that Nora achieves superior performance in both convergence speed and wall-clock execution time. Our contributions are three-fold:

- We propose Nora, a novel optimizer that simultaneously achieves efficiency in preconditioning estimation, stability by respecting the scale invariance of neural networks, and high computational speed during training.
- We evaluated Nora by training LLaMA models across various learning rates. Extensive experiments demonstrate that our algorithm achieves better results in terms of both training efficiency and computational wall-clock time.
- We rigorously prove through theorems that Nora is a scalable optimizer and provide scaling criteria. We also provide rigorous convergence guarantees for Nora in non-convex optimization, demonstrating that our approach is characterized by theoretical completeness.

2 Notations

Throughout this paper, $w \in \mathbb{R}^{m\times n}$ denotes the learnable matrix parameters, and $f(w)$ denotes the loss function. Let w_t, g_t, and v_t denote the parameters, gradients, and momentum at iteration t, respectively. The operator $\mathrm{diag}(\cdot)$ keeps the diagonal entries of a square matrix and sets all off-diagonal entries to zero. We denote the (i,j)-th entry of the momentum matrix by $v_{t,ij}$ and its i-th row by $v_{t,i:}$. We decompose momentum into $v^{\parallel}$ and $v^{\perp}$, the components parallel and orthogonal to w. Similarly, $v^{r\parallel}$ and $v^{r\perp}$ denote row-wise parallel and orthogonal components, while $v^{c\parallel}$ and $v^{c\perp}$ denote the corresponding column-wise components. The symbol $\otimes$ denotes the Kronecker product, and $\odot$ and $\oslash$ denote element-wise multiplication and division. The notation $(\cdot)^T$ denotes matrix transpose, and $H^{-1}[\cdot]$ denotes the action of a preconditioning operator on a vector or matrix.

3 Related Work

3.1 Matrix-Based Optimizers and Muon

Adaptive optimizers such as AdamW [24] scale gradients with diagonal preconditioners and therefore ignore off-diagonal curvature in the loss landscape $f(w)$. Matrix-based optimizers instead treat the

weight $w \in \mathbb{R}^{m\times n}$, typically with $m < n$, as a structured matrix [25]. Let $g_t = \nabla f(w_t)$, and let $v_t = \beta v_{t-1} + (1-\beta)g_t$ denote the exponential moving average of the gradient. Muon orthogonalizes this momentum and uses the update direction $d_t^{\mathrm{M}} = (v_t v_t^T)^{-\frac{1}{2}} v_t$, thereby applying a global structural preconditioner. To avoid the exact inverse square root, Muon uses the Newton–Schulz iteration $X_{k+1} = \frac{1}{2}X_k(3I - X_k^T X_k)$ from a scaled initialization $X_0 \propto v_t$. This update improves the conditioning of the search direction, but the dense matrix multiplications in the Newton–Schulz loop cost $\mathcal{O}(m^2 n)$ [26], limiting scalability in large LLM layers.

3.2 Row-Momentum Normalized Preconditioning

Recent studies show that the layer-wise Hessian matrices of Transformers exhibit strong row-wise block-diagonal dominance [27]. Muon can be written as applying the full-spectral preconditioner $H_M = (v_t v_t^T)^{\frac{1}{2}} \otimes I_n$, where v_t is the momentum matrix. Row-Momentum Normalized Preconditioning (RMNP) uses the row-dominant structural prior by retaining only the diagonal blocks of the Gram matrix: $H_R = (\mathrm{diag}(v_t v_t^T))^{\frac{1}{2}} \otimes I_n$. This approximation reduces matrix orthogonalization to row-wise ℓ_2 normalization of the momentum: $H_R^{-1}[v_t]_{i:} = \frac{v_{t,i:}}{\|v_{t,i:}\|_2}$. By eliminating the Newton–Schulz loop, RMNP reduces the per-iteration complexity from $\mathcal{O}(mn \cdot \min(m,n))$ to $\mathcal{O}(mn)$. However, this algebraic simplification neglects the scale symmetry of neural networks and allows unconstrained radial updates that inject noise into the training dynamics.

3.3 Scale-invariance in Neural Networks

The widespread use of BatchNorm, RMSNorm, and LayerNorm makes scale invariance a central property of modern neural networks [28]. For scale-invariant parameters w, the loss satisfies $f(w) = f(\lambda w)$ for any $\lambda > 0$. This property implies that the gradient is orthogonal to the weights: $\langle \nabla f(w), w \rangle = 0$. Thus, the magnitude of w is decoupled from the loss value, and learning changes the angular orientation of the parameters [29, 30]. In this setting, the radial momentum component $v^{\parallel}$ interferes with angular optimization and perturbs the weight norm. This perturbation destabilizes the effective learning rate, defined as the relative angular step size, and hinders stable training [31]. Our goal is therefore to update along directions d_t satisfying $\langle d_t, w_t \rangle = 0$, so that the update remains orthogonal to w_t.

4 Nora Optimizer

4.1 Design Concept

Modern neural networks contain many scale-invariant parameters: changing $\|w_t\|_F$ does not change the loss. The useful motion is therefore angular rather than radial. **Based on the principle of stability**, the angular velocity should remain controlled by the learning-rate schedule. Because preconditioning typically produces an update magnitude determined by the matrix dimensions $\mathbb{R}^{m\times n}$, the preconditioned direction must remain perpendicular to w_t. Otherwise, radial components change the weight norm and the effective learning rate. Tangential updates keep norm growth controlled and allocate momentum to directions that affect the network output.

Existing preconditioners do not satisfy this requirement. Even when applied to tangential momentum $v_t^{\perp}$, the Muon operator produces an update $d_t^M = (v_t^{\perp}(v_t^{\perp})^T)^{-\frac{1}{2}} v_t^{\perp}$ that deviates from the tangent direction, so $\langle d_t^M, w_t \rangle \neq 0$. RMNP has the same issue: its preconditioner can destroy the orthogonality of the update to w_t. SSO [32] preserves this geometric property, but it relies on a computationally intensive bisection method and is too slow in practice.

Inspired by prior work on the row-diagonal dominance of the Transformer Hessian, we observe that preconditioning can be made compatible with tangential updates by considering $v_t^{r\perp}$. Here, $v_t^{r\perp}$ is obtained by orthogonalizing v_t against each row of w_t, ensuring that $\langle v_{t,i:}^{r\perp}, w_{t,i:} \rangle = 0$ for every row i. **Based on the principle of efficiency**, we seek a preconditioner for $v_t^{r\perp}$. Given the row-diagonal dominance of the Transformer Hessian, it follows that $v_t^{r\perp}(v_t^{r\perp})^T$ also exhibits row-diagonal dominance. We therefore adopt the diagonal elements of $v_t^{r\perp}(v_t^{r\perp})^T$ as the preconditioner, specifically $H_N = (\mathrm{diag}(v_t^{r\perp}(v_t^{r\perp})^T))^{\frac{1}{2}} \otimes I_n$. Consequently, the preconditioned update simplifies

to $H_N^{-1}[v_t^{r\perp}] = (\text{diag}(v_t^{r\perp}(v_t^{r\perp})^T))^{-\frac{1}{2}} v_t^{r\perp}$. Through algebraic simplification, we arrive at:

$$H_N^{-1}[v_t^{r\perp}]_{i:} = ((\text{diag}(v_t^{r\perp}(v_t^{r\perp})^T))^{-\frac{1}{2}} v_t^{r\perp})_{i:} = \frac{v_{t,i:}^{r\perp}}{\|v_{t,i:}^{r\perp}\|_2} \tag{1}$$

In other words, applying the preconditioner $H_N[\cdot]$ to $v_t^{r\perp}$ is mathematically equivalent to performing a simple row-wise normalization. Furthermore, the following Equation (2) leads to our long-sought objective: by preconditioning $v_t^{r\perp}$ (which is already row-orthogonal to w_t), we simultaneously achieve optimal efficiency while preserving the orthogonality between $v_t^{r\perp}$ and w_t. This ensures that the update consistently adheres to the principle of stability.

$$\begin{aligned}
&\langle H_N^{-1}[v_t^{r\perp}], w_t\rangle = \sum_{i=1}^{m} \langle H_N^{-1}[v_t^{r\perp}]_{i:}, w_{t,i:}\rangle \\
&= \sum_{i=1}^{m} \langle \frac{1}{\|v_{t,i:}^{r\perp}\|_2} v_{t,i:}^{r\perp}, w_{t,i:}\rangle = \frac{1}{\|v_{t,i:}^{r\perp}\|_2} \sum_{i=1}^{m} \langle v_{t,i:}^{r\perp}, w_{t,i:}\rangle \\
&= \langle (H_N^{-1}[v_t^{r\perp}])^{\perp}, w_t\rangle = 0
\end{aligned} \tag{2}$$

From the perspective of Speed, Nora maintains high efficiency and stability while introducing minimal computational overhead. The additional computation consists solely of row-wise projections of v_t onto the tangent space of w_t, which are element-wise operations that consume virtually no additional time compared to RMNP. Furthermore, by exploiting the diagonal dominance of the Transformer Hessian, Nora replaces the computationally intensive Newton-Schulz (NS) iterations used in Muon with simple row-wise normalization. This effectively reduces the overall time complexity and ensures optimal training throughput.

Furthermore, updating along the direction of $(\text{diag}(v_t^{r\perp}(v_t^{r\perp})^T))^{-\frac{1}{2}} v_t^{r\perp}$ offers additional advantages. First, the discrete tangential updates ensure that the norm of each row increases only at a second-order rate. This growth is both monotonic—guaranteeing steady progression without erratic fluctuations—and gentle, remaining strictly controllable throughout training. **Moreover**, the row-wise orthogonality of d_t relative to w_t ensures a uniform growth rate across all rows of w_t. By upholding the geometric consistency of scale-invariant parameters, this mechanism effectively mitigates internal covariate shift and maintains the representation balance of the neural network.

The synthesis of these three properties forms the core design philosophy behind Nora, ensuring that it inherently satisfies the requirements for training efficiency, stability, and speed. The comprehensive procedure of the Nora algorithm is detailed in Algorithm 1. Furthermore, Table 1 presents a comparative summary of how various optimizers adhere to the three core design principles.

Algorithm 1: The Nora Optimizer

Require: Layer Weight $w_t \in \mathbb{R}^{m\times n}$, momentum $v_t \in \mathbb{R}^{m\times n}$, learning rate η_t at step t, momentum coefficient β, and weight decay coefficient λ (Default λ=0).
Initialize $v_0 \leftarrow \mathbf{0} \in \mathbb{R}^{m\times n}, t \leftarrow 0$.
for each step **do**
  $g_t \leftarrow \nabla f(w_t)$
  $v_t \leftarrow \beta v_{t-1} + (1-\beta) g_t$
  $v_{t,i:}^{r\perp} \leftarrow v_{t,i:} - \frac{\langle v_{t,i:}, w_{t,i:}\rangle}{\|w_{t,i:}\|_2^2} w_{t,i:}$ {For all Row}
  $d_t \leftarrow (\text{diag}(v_t^{r\perp}(v_t^{r\perp})^T))^{-\frac{1}{2}} v_t^{r\perp}$ {Row Normalization}
  $w_{t+1} \leftarrow w_t - \eta_t(d_t + \lambda\theta_t)$
end for

4.2 Theoretical Analysis

We now analyze Nora as a scalable optimizer and establish non-convex convergence guarantees. For a matrix $x \in \mathbb{R}^{m\times n}$, define:

$$\|x\|_{1,2} := \sum_{i=1}^{m} \|x_{i:}\|_2, \qquad \|x\|_{\infty,2} := \max_{1\le i\le m} \|x_{i:}\|_2.$$

Table 1: Comparison of Optimizers Based on the Three Design Principles

Optimizer	Nora	Muon [4]	RMNP [7]	SSO [32]	SGD [33]	Adam [1]	AdamP [16]
Efficiency	✓	✓	✓	✓	✗	✗	✗
Stability	✓	✗	✗	✓	✗	✗	✓
Speed	✓	✗	✓	✗	✓	✓	✓

For any matrix w with nonzero rows, a matrix z is row-wise perpendicular to w if $\langle z_{i:}, w_{i:}\rangle = 0$ for every row i. The row-wise perpendicular projection is:

$$[\mathcal{P}_w^{r\perp}(x)]_{i:} := x_{i:} - \frac{\langle x_{i:}, w_{i:}\rangle}{\|w_{i:}\|_2^2}\, w_{i:}. \tag{3}$$

We use row-wise normalization with the convention $0/0 = 0$:

$$[\mathrm{RN}(x)]_{i:} := \begin{cases} \dfrac{x_{i:}}{\|x_{i:}\|_2}, & x_{i:} \neq 0, \\ 0, & x_{i:} = 0. \end{cases} \tag{4}$$

Throughout this subsection, the scalar m denotes the number of rows of $w \in \mathbb{R}^{m\times n}$, and v_t denotes the momentum sequence. For $t = 0, \ldots, T-1$, we analyze the core Nora update without decoupled weight decay:

$$\begin{aligned} g_t &= \nabla f(w_t; \xi_t), \quad v_{t+1} = \beta v_t + (1-\beta) g_t, \qquad v_0 = 0, \\ v_{t+1}^{r\perp} &= \mathcal{P}_{w_t}^{r\perp}(v_{t+1}), \qquad d_t = \mathrm{RN}(v_{t+1}^{r\perp}), \qquad w_{t+1} = w_t - \eta d_t. \end{aligned} \tag{5}$$

4.2.1 Scaling for Nora

In this section, we address the most fundamental engineering question in LLM training: How should the learning rate η of Nora scale with the model width? According to the core principles of Maximal Update Parametrization (μP), an optimal optimizer must ensure that the change in hidden layer activations, Δh, remains on the scale of $\Theta(1)$ as the network width increases ($n \to \infty$). This ensures the network neither fails to learn features due to vanishing updates nor suffers from numerical explosion. We derive the rigorous learning rate formula required for Nora to satisfy this limit through the following theorem:

Theorem 4.1 (Nora Scaling under the Scaling Hypothesis)**.** *Consider a neural network layer defined by $h = wx$, where the weights $w \in \mathbb{R}^{m\times n}$. Assume the input activation $x \in \mathbb{R}^n$ satisfies the scaling hypothesis under standard deep learning initialization: $\|x\|_2 \le \gamma\sqrt{n}$, where $\gamma = \Theta(1)$ is a constant. Suppose the parameters are updated using Nora: $w_{t+1} = w_t - \eta_t d_t$. Then $|\Delta h_i| \le \eta_t \gamma \sqrt{n}$. To achieve the stable feature learning limit required by μP theory—specifically, to ensure the activation update magnitude satisfies $|\Delta h_i| = \Theta(1)$—the learning rate η_t for Nora must follow the width-scaling rule, $\eta_t \propto 1/\sqrt{n}$.*

The theorem provides an upper bound on the forward update, $|\Delta h_i| \le \eta_t \gamma \sqrt{n}$. This bound ensures training stability but raises a more precise question: do Nora's projection and normalization operations substantially weaken the effective gradient components in practice, causing Δh_i to fall significantly below $\sqrt{n}$ or even vanish? To show that Nora achieves Maximal Update Parametrization (μP), we must evaluate the true order of Δh_i in the high-dimensional limit $n \to \infty$. To this end, we use the following theorem.

Theorem 4.2 (Asymptotic Convergence)**.** *Consider a hidden layer $h = wx$, where $w \in \mathbb{R}^{m\times n}$. Under the standard infinite-width random initialization hypothesis, assume the row vectors of w follow $w_{i:} \sim \mathcal{N}(0, \sigma_w^2 I_n/n)$, and the components of the input activation $x \in \mathbb{R}^n$ are independent with zero mean and variance $\sigma_x^2 = \Theta(1)$. Let the error signal backpropagated to this layer be $\delta_i = \Theta(1)$, and let the vanilla gradient without momentum be $g_{i:} = \delta_i x^\top$. If Nora generates the update direction $d_{i:} = \mathrm{RN}(\mathcal{P}_w^{r\perp}(g_{i:}))$, then as the network width $n \to \infty$, the inner product between the update direction and the input converges in probability to $\langle d_{i:}, x\rangle \xrightarrow{p} \mathrm{sgn}(\delta_i)\sigma_x\sqrt{n}$. Therefore, to obtain non-trivial feature learning with $\Delta h_i = \Theta(1)$, Nora must use $\eta = \eta_0/\sqrt{n}$.*

4.2.2 Convergence Analysis

We next prove non-convex convergence guarantees for Nora. Define the natural filtration as:

$$\mathcal{F}_t := \sigma(w_0, \xi_0, \ldots, \xi_{t-1}).$$

It is the sigma-algebra generated before sampling ξ_t. Equivalently, w_t and v_t are $\mathcal{F}_t$-measurable, and the conditional expectations below are taken only over the current stochastic gradient. We measure stationarity by the row-wise projected gradient:

$$\mathcal{G}_t := \mathcal{P}_{w_t}^{r\perp}\big(\nabla f(w_t)\big). \tag{6}$$

This is the relevant first-order signal for Nora because each row of d_t is perpendicular to the corresponding row of w_t. Hence its descent inner product depends only on the row-wise perpendicular component of the true gradient, not on the radial component removed by the projection. Under row-wise scale invariance, $\mathcal{G}_t$ coincides with $\nabla f(w_t)$, so the projected stationarity measure reduces to the standard first-order one.

Compared with RMNP, Nora inserts one row-wise perpendicular projection before row-wise normalization. This additional step preserves the RMNP proof structure; the main new ingredient is the non-expansiveness of $\mathcal{P}_{w_t}^{r\perp}$ in the norms used below.

Assumption 4.3 (Smoothness)**.** The objective $f : \mathbb{R}^{m\times n} \to \mathbb{R}$ satisfies one of the following conditions:

(a) (Frobenius smoothness) There exists $L_F > 0$ such that, for all w, w',

$$\|\nabla f(w) - \nabla f(w')\|_F \le L_F \|w - w'\|_F. \tag{7}$$

(b) (Matched $(\infty, 2)$-smoothness) There exists $L_{\infty,2} > 0$ such that, for all w, w',

$$\|\nabla f(w) - \nabla f(w')\|_{1,2} \le L_{\infty,2} \|w - w'\|_{\infty,2}. \tag{8}$$

Theorem 4.4 (Nora under matched $(\infty, 2)$-smoothness)**.** *Suppose Assumptions 4.3(b) hold. If Nora uses a constant step size $\eta_t = \eta$, then:*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\big[\|\mathcal{G}_t\|_{1,2}\big] \le \frac{\Delta}{T\eta} + 2\left[\left(1 - \frac{1}{T}\right)\frac{L_{\infty,2}\eta\beta}{1-\beta} + \frac{\sqrt{m}\sigma}{\sqrt{B}}\sqrt{\frac{1-\beta}{1+\beta}}\right] + \frac{L_{\infty,2}\eta}{2}. \tag{9}$$

Here m is the row dimension of w. With the choice:

$$\eta = \sqrt{\frac{(1-\beta)\Delta}{L_{\infty,2}T}}, \qquad 1 - \beta = \min\left\{\frac{\sqrt{L_{\infty,2}\Delta}}{2\sqrt{m}\sigma\sqrt{T}}, 1\right\}, \tag{10}$$

Nora reaches an ϵ-stationary point in the projected $\|\cdot\|_{1,2}$ sense after:

$$T = \mathcal{O}\big(mL_{\infty,2}\sigma^2\Delta\,\epsilon^{-4}\big) \tag{11}$$

iterations.

Proposition 4.5 (Frobenius-smooth counterparts)**.** *Under the same conditions of Theorem 4.4, if Nora uses a constant step size $\eta_t = \eta$, then:*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\big[\|\mathcal{G}_t\|_F\big] \le \frac{\Delta}{T\eta} + (\sqrt{m}+1)\left[\left(1 - \frac{1}{T}\right)\frac{L_F\eta\sqrt{m}\beta}{1-\beta} + \frac{\sigma}{\sqrt{B}}\sqrt{\frac{1-\beta}{1+\beta}}\right] + \frac{L_F\eta m}{2}. \tag{12}$$

The corresponding projected $\|\cdot\|_{1,2}$ bound is:

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\big[\|\mathcal{G}_t\|_{1,2}\big] \le \frac{\Delta}{T\eta} + 2\left[\left(1 - \frac{1}{T}\right)\frac{L_F\eta m\beta}{1-\beta} + \frac{\sqrt{m}\sigma}{\sqrt{B}}\sqrt{\frac{1-\beta}{1+\beta}}\right] + \frac{L_F\eta m}{2}. \tag{13}$$

Consequently, Nora reaches an ϵ-stationary point in either projected measure after:

$$T = \mathcal{O}\big(m^2L_F\sigma^2\Delta\,\epsilon^{-4}\big) \tag{14}$$

iterations.

Corollary 4.6 (Standard first-order stationarity under row-wise scale invariance)**.** *Assume, in addition, that f is row-wise scale invariant:*

$$f(Dw) = f(w), \qquad \forall w \in \mathbb{R}^{m\times n},\ \forall D \succ 0 \text{ diagonal}. \tag{15}$$

Then $\nabla f(w_t)$ is row-wise perpendicular to w_t for all t, i.e.,

$$\mathcal{P}_{w_t}^{r\perp}\big(\nabla f(w_t)\big) = \nabla f(w_t). \tag{16}$$

Therefore, Theorem 4.4 and Proposition 4.5 hold with $\mathcal{G}_t$ replaced by $\nabla f(w_t)$.

Proof sketch. The proof follows the RMNP descent argument and adds one projection lemma: for fixed w_t, the row-wise perpendicular projector $\mathcal{P}_{w_t}^{r\perp}$ is non-expansive in $\|\cdot\|_F$, $\|\cdot\|_{1,2}$, and $\|\cdot\|_{\infty,2}$. Because each row of d_t is perpendicular to the corresponding row of w_t,

$$\langle \nabla f(w_t), d_t \rangle = \left\langle \mathcal{P}_{w_t}^{r\perp}(\nabla f(w_t)), d_t \right\rangle = \langle \mathcal{G}_t, d_t \rangle. \tag{17}$$

Define the projected momentum tracking error as:

$$e_t := v_{t+1}^{r\perp} - \mathcal{G}_t.$$

The row-normalization identities give:

$$\langle \mathcal{G}_t, d_t \rangle \geq \|\mathcal{G}_t\|_{1,2} - 2\|e_t\|_{1,2}, \qquad \langle \mathcal{G}_t, d_t \rangle \geq \|\mathcal{G}_t\|_F - (\sqrt{m}+1)\|e_t\|_F. \tag{18}$$

Combining these inequalities with smoothness-based descent and the standard momentum recursion yields the stated bounds. Full proofs and standard assumptions are deferred to Appendix A.

4.3 Compared with Mano

In this section, we delineate the distinctions between Mano [34] and Nora. Mano alternates between row-wise and column-wise perpendicular projections, followed by normalization. Specifically, Mano accumulates momentum v_t in the standard manner. During odd iterations, it removes the row-wise radial component to obtain $v_t^{r\perp}$ and applies row normalization: $d_t = \text{RN}(v_t^{r\perp})$. During even iterations, it removes the column-wise radial component to obtain $v_t^{c\perp}$ and applies column normalization: $d_t = \text{CN}(v_t^{c\perp})$.

The design philosophy of Nora diverges significantly from that of Mano. Nora originates from the properties expected by the optimizer itself, while Mano comes from Riemannian optimization [35, 36, 37]. In practical implementation, Nora is more streamlined as it eliminates the need to track iteration parity (odd vs. even steps). More importantly, Nora is explicitly designed to leverage the block-diagonal dominance characteristic of the Transformer Hessian. By removing the heuristic column-wise orthogonal projection and normalization used in Mano, Nora demonstrates superior properties in ablation studies. These empirical results suggest that the diagonal dominance of the Transformer Hessian is a critical structural prior that must be prioritized in LLM optimizer design.

5 Experiments

We evaluate Nora on autoregressive language modeling with LLaMA-style Transformer models at two scales: 60M and 135M parameters. We compare Nora with three matrix-based optimizers, Muon, Mano, and RMNP, under the same data pipeline, tokenizer, context length, global batch size, learning-rate schedule, precision, evaluation cadence, and checkpointing cadence. Detailed model and training configurations are deferred to Appendix B.

The compared optimizers differ only in the optimizer rule, the matrix learning-rate sweep grid, and weight decay. For all non-matrix parameter groups, we use the same auxiliary Adam setting within each model size. For matrix-shaped parameters, we tune the matrix learning rate over optimizer-specific grids and select the best run by validation loss. Muon uses a slightly different sweep range from the other matrix optimizers because it is more sensitive to large matrix learning rates in our preliminary runs. The full sweep grids are listed in Appendix B. Importantly, Nora uses weight decay 0 in all runs, whereas Muon, Mano, and RMNP use weight decay 0.1. We treat this as part of Nora's optimizer configuration and report it explicitly in all result tables.

5.1 Best Results from Different Optimizers

Table 2 reports the main language-modeling results. For each optimizer and model size, we report the matrix learning rate selected from the sweep, the validation loss at the selected setting, and validation perplexity. This format separates peak validation performance from the final checkpoint quality.

Figure 1 shows the 135M training dynamics from 10k to 20k steps. RMNP has the strongest early-stage loss and perplexity, while Muon improves more slowly and plateaus at a higher value. Mano and Nora start from higher loss and perplexity, but both continue to improve after 15k steps. Nora shows the clearest late-stage improvement: it overtakes the other methods near the end of training

Table 2: Main language-modeling results. For each optimizer and model size, the matrix learning rate is selected by validation loss over the sweep grid in Appendix B. Lower validation loss and perplexity are better. Best results are shown in bold.

Optimizer	Weight decay	60M			135M		
		Best matrix LR	Val. loss	Val. ppl.	Best matrix LR	Val. loss	Val. ppl.
Muon	0.1	0.01	3.44	31.09	0.01	3.142	23.17
Mano	0.1	0.004	3.39	29.55	0.003	3.097	22.13
RMNP	0.1	0.004	3.41	30.12	0.01	3.112	22.46
Nora	0.0	0.004	**3.37**	**28.94**	0.003	**3.079**	**21.74**

Table 3: Comparison of Mano and Nora (Model: 135M, $wd = 0$)

Optimizer	0.003		0.005		0.01		0.02	
	ppl	loss	ppl	loss	ppl	loss	ppl	loss
Mano	22.13	3.097	22.14	3.098	23.34	3.150	25.07	3.222
Nora	21.74	3.079	21.86	3.085	22.43	3.111	23.39	3.152

and reaches the lowest final loss and perplexity. This indicates that Nora uses the same training budget more **efficiently**, achieving better optimization quality without requiring additional training steps. This trend is consistent with Table 2, where Nora achieves the best 135M validation loss and perplexity among the compared optimizers.

5.2 Runtime Comparison

We also compare the cost of row normalization with Newton–Schulz orthogonalization on representative matrix shapes from LLaMA-style models. The benchmark uses CUDA with bfloat16 precision, 20 warmup iterations, and 200 measured iterations. Newton–Schulz uses five iterations. As shown in Table 4, row normalization is consistently cheaper than Newton–Schulz. The gap is already about one order of magnitude for small and medium matrices, and becomes much larger for the 1B-scale MLP matrices, where Newton–Schulz is more than 70 times slower. This highlights the substantial **speed** advantage of row normalization in practical training settings. Results support the practical motivation for replacing matrix-level orthogonalization with a row-wise normalization step.

Table 4: Runtime comparison between row normalization and five-step Newton–Schulz orthogonalization on representative LLaMA-style matrix shapes. Times are mean kernel runtimes in milliseconds under CUDA and bfloat16 precision.

Model scale	Matrix shape	Representative layer	Row normalization (ms)	NS(5) (ms)	NS / row-norm
60M	512×512	attention: hidden $\times$ hidden	0.0689	0.6554	9.52$\times$
60M	1376×512	MLP: intermediate $\times$ hidden	0.0682	0.6853	10.05$\times$
60M	512×1376	MLP: hidden $\times$ intermediate	0.0688	0.6623	9.63$\times$
135M	768×768	attention: hidden $\times$ hidden	0.0686	0.6520	9.50$\times$
135M	2048×768	MLP: intermediate $\times$ hidden	0.0692	0.6871	9.93$\times$
135M	768×2048	MLP: hidden $\times$ intermediate	0.0691	0.6534	9.46$\times$
350M	1024×1024	attention: hidden $\times$ hidden	0.0674	0.6397	9.49$\times$
350M	2816×1024	MLP: intermediate $\times$ hidden	0.0687	0.8625	12.56$\times$
350M	1024×2816	MLP: hidden $\times$ intermediate	0.0675	0.6941	10.28$\times$
1B	2048×2048	attention: hidden $\times$ hidden	0.0684	2.0552	30.06$\times$
1B	5461×2048	MLP: intermediate $\times$ hidden	0.0985	6.9985	71.02$\times$
1B	2048×5461	MLP: hidden $\times$ intermediate	0.1084	7.9678	73.51$\times$

5.3 Ablation Experiment between Nora and Mano

To verify that row-normalization indeed captures the row-diagonal dominance inherent in the Transformer Hessian, we conducted an ablation study. Given that Nora and Mano share similar algorithmic frameworks, it is crucial to distinguish their structural advantages from hyperparameter effects. While the previous experiments utilized Mano's recommended $wd = 0.1$ against Nora's default $wd = 0$, we eliminated this discrepancy in the ablation study by setting Mano's wd to 0. As shown in Table 3,

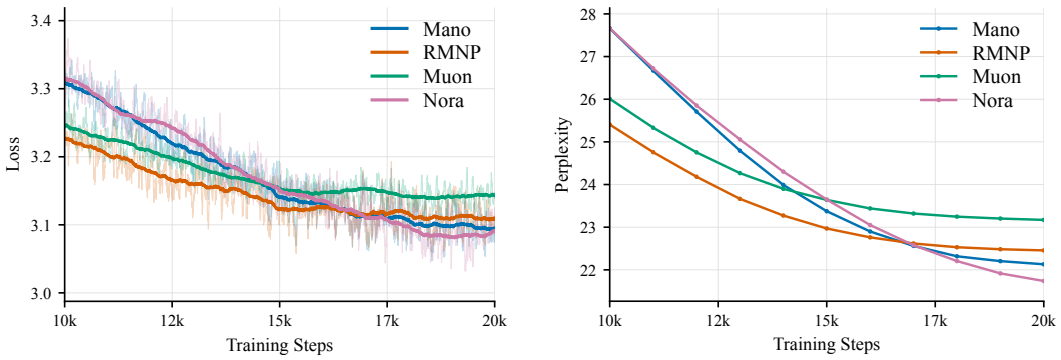

Figure 1: Training dynamics on the 135M model. Left: loss over training steps. Right: perplexity over training steps. Nora continues to improve late in training and finishes with the lowest loss and perplexity.

the results demonstrate that Nora's superiority stems from its intrinsic algorithmic architecture rather than an advantage gained from weight decay settings.

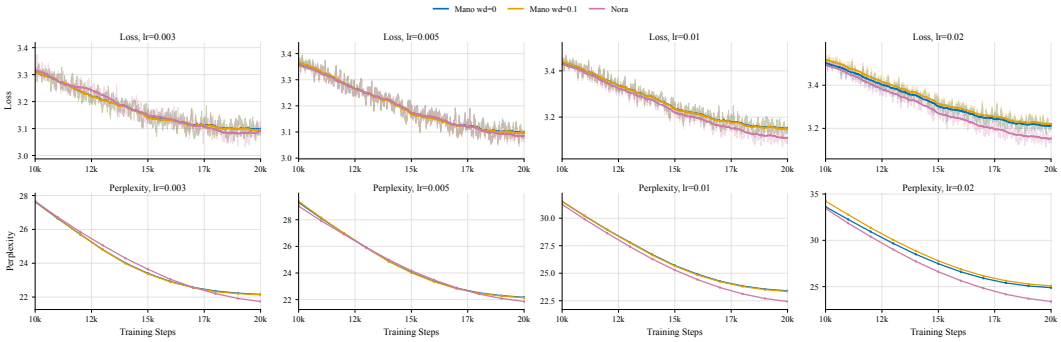

Figure 2: Training dynamics on the 135M model. This figure illustrates the perplexity and loss decay curves for Nora (default $wd = 0$) in comparison with Mano (under $wd = 0$ and $wd = 0.1$).

Furthermore, Figure 2 illustrates the perplexity and loss decay curves for both Mano and Nora at the 135M model scale, under configurations of $wd = 0$ and $wd = 0.1$. It is evident that Nora consistently outperforms Mano across all settings. For a comprehensive overview, all remaining experimental results and corresponding convergence curves are provided in Appendix B, where we focus on the sensitivity of hyper-parameters and **stability** of our proposed optimizer.

6 Conclusion

We introduce Nora, a normalized orthogonal row-alignment optimizer for scalable LLM training. Nora is motivated by the geometry of scale-invariant neural networks, where effective learning should mainly occur along angular rather than radial directions. By projecting momentum onto the row-wise orthogonal complement of the weights and applying row-wise normalization, Nora preserves stable scale-invariant dynamics while retaining efficient Muon-like preconditioning under the row-diagonal structure of Transformer Hessians. We further establish Nora's scalability through μP-based width-scaling analysis and provide non-convex convergence guarantees. Experiments on LLaMA-style models show that Nora achieves the best validation loss and perplexity among the compared matrix-based optimizers, while its row-wise normalization brings a clear speed advantage over Newton–Schulz orthogonalization. Overall, Nora demonstrates that efficiency, stability, and speed can be unified through a simple row-wise geometric principle, offering a practical and principled direction for scalable optimizer design.

References

[1] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[2] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[3] Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025.

[4] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024.

[5] Günther Schulz. Iterative berechung der reziproken matrix. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 13(1):57–59, 1933.

[6] Zhaorui Dong, Yushun Zhang, Jianfeng Yao, and Ruoyu Sun. Towards quantifying the hessian structure of neural networks. In *OPT 2025: Optimization for Machine Learning*.

[7] Shenyang Deng, Zhuoli Ouyang, Tianyu Pang, Zihang Liu, Ruochen Jin, Shuhua Yu, and Yaoqing Yang. Rmnp: Row-momentum normalized preconditioning for scalable matrix-based optimization. *arXiv preprint arXiv:2603.20527*, 2026.

[8] Johanni Brea, Berfin Simsek, Bernd Illing, and Wulfram Gerstner. Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape. *arXiv preprint arXiv:1907.02911*, 2019.

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[10] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

[11] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[12] Ruosi Wan, Zhanxing Zhu, Xiangyu Zhang, and Jian Sun. Spherical motion dynamics: Learning dynamics of normalized neural network using sgd and weight decay. *Advances in Neural Information Processing Systems*, 34:6380–6391, 2021.

[13] Jh Yuan and Feiping Nie. Spherical cautious optimizers. In *Workshop on Scientific Methods for Understanding Deep Learning*, 2026.

[14] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016.

[15] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International conference on machine learning*, pages 4596–4604. PMLR, 2018.

[16] Byeongho Heo, Sanghyuk Chun, Seong Joon Oh, Dongyoon Han, Sangdoo Yun, Gyuwan Kim, Youngjung Uh, and Jung-Woo Ha. Adamp: Slowing down the slowdown for momentum optimizers on scale-invariant weights. *arXiv preprint arXiv:2006.08217*, 2020.

[17] Hao Chen, Jh Yuan, and Hanmin Zhang. Decoupled orthogonal dynamics: Regularization for deep network optimizers. In *Workshop on Scientific Methods for Understanding Deep Learning*, 2026.

[18] Xingyu Xie, Pan Zhou, Huan Li, Zhouchen Lin, and Shuicheng Yan. Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9508–9520, 2024.

[19] Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar C Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *Advances in neural information processing systems*, 33:18795–18806, 2020.

[20] Hong Liu, Zhiyuan Li, David Leo Wright Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *The Twelfth International Conference on Learning Representations*, 2024.

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

[22] Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. *Advances in Neural Information Processing Systems*, 34:17084–17097, 2021.

[23] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[24] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[25] Kaiyue Wen, David Hall, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them. *arXiv preprint arXiv:2509.02046*, 2025.

[26] Nicholas J Higham. *Functions of matrices: theory and computation*. SIAM, 2008.

[27] Yushun Zhang, Congliang Chen, Tian Ding, Ziniu Li, Ruoyu Sun, and Zhi-Quan Luo. Why transformers need adam: A hessian perspective. *Advances in neural information processing systems*, 37:131786–131823, 2024.

[28] Behnam Neyshabur, Russ R Salakhutdinov, and Nati Srebro. Path-sgd: Path-normalized optimization in deep neural networks. *Advances in neural information processing systems*, 28, 2015.

[29] Kaiyue Wen, Xingyu Dang, Kaifeng Lyu, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them 2.1: Hyperball optimization, 12 2025.

[30] Liliang Ren, Yang Liu, Yelong Shen, and Weizhu Chen. Rethinking language model scaling under transferable hypersphere optimization. *arXiv preprint arXiv:2603.28743*, 2026.

[31] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*, 2020.

[32] Tian Xie, Haoming Luo, Haoyu Tang, Yiwen Hu, Jason Klein Liu, Qingnan Ren, Yang Wang, Wayne Xin Zhao, Rui Yan, Bing Su, et al. Controlled llm training on spectral sphere. *arXiv preprint arXiv:2601.08393*, 2026.

[33] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[34] Yufei Gu and Zeke Xie. Mano: Restriking manifold optimization for llm training, 2026.

[35] Nicolas Boumal. *An introduction to optimization on smooth manifolds*. Cambridge University Press, 2023.

[36] Jh Yuan, Fangyuan Xie, Feiping Nie, and Xuelong Li. Riemannian optimization on relaxed indicator matrix manifold. In *The Fourteenth International Conference on Learning Representations*, 2026.

[37] Jh Yuan, Zhuo Liu, and Feiping Nie. Riemannian fuzzy k-means on product manifolds. In *Non-Euclidean Foundation Models: Advancing AI Beyond Euclidean Frameworks*, 2025.

[38] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Contents

A Proof of Theorem

A.1 Proof of Theorem 4.1

Theorem 4.1 (Nora Scaling under the Scaling Hypothesis) *Consider a neural network layer defined by $h = wx$, where the weights $w \in \mathbb{R}^{m\times n}$. Assume the input activation $x \in \mathbb{R}^n$ satisfies the scaling hypothesis under standard deep learning initialization: $\|x\|_2 \le \gamma\sqrt{n}$, where $\gamma = \Theta(1)$ is a constant. Suppose Nora updates the parameters by $w_{t+1} = w_t - \eta_t d_t$. Then $|\Delta h_i| \le \eta_t\gamma\sqrt{n}$. To achieve the stable feature learning limit required by μP theory—specifically, to ensure the activation update magnitude satisfies $|\Delta h_i| = \Theta(1)$, the learning rate η_t for Nora must follow the width-scaling rule, $\eta_t \propto \frac{1}{\sqrt{n}}$.*

Proof. For any output neuron i, the change in the hidden activation Δh_i during the forward pass is:

$$\Delta h_i = (w_{t+1,i:} - w_{t,i:})x = -\eta_t\langle d_{t,i:}, x\rangle \tag{19}$$

By taking the absolute value and applying the Cauchy-Schwarz inequality, we obtain:

$$|\Delta h_i| = \eta_t|\langle (d_t)_{i:}, x\rangle| \le \eta_t\|(d_t)_{i:}\|_2\|x\|_2 \tag{20}$$

Nora's row-wise normalization strictly ensures $\|(d_t)_{i:}\|_2 \le 1$. Combined with the input assumption $\|x\|_2 \le \gamma\sqrt{n}$, substituting these into the inequality yields a rigorous upper bound for the forward update:

$$|\Delta h_i| \le \eta_t \cdot 1 \cdot \gamma\sqrt{n} = \eta_t\gamma\sqrt{n} \tag{21}$$

To ensure that the feature update magnitude $|\Delta h_i|$ neither diverges nor vanishes as $n \to \infty$ (i.e., satisfies the Maximal Update condition $|\Delta h_i| = \Theta(1)$), we require the upper bound $\eta_t\gamma\sqrt{n}$ to also be of order $\Theta(1)$. Given that γ is a constant, this necessarily implies $\eta_t = \frac{c}{\sqrt{n}}$, where c is a constant independent of n. This completes the proof.

A.2 Proof of Theorem 4.2

Theorem 4.2 (symptotic Convergence)) *Consider a hidden layer $h = wx$, where $w \in \mathbb{R}^{m\times n}$. Under the standard infinite-width random initialization hypothesis, assume the row vectors of w follow $w_{i:} \sim \mathcal{N}(0, \frac{\sigma_w^2}{n}I_n)$, and the components of the input activation $x \in \mathbb{R}^n$ are independent with zero mean and variance $\sigma_x^2 = \Theta(1)$. Let the error signal backpropagated to this layer be $\delta_i = \Theta(1)$, and the vanilla gradient (without momentum) be $g_{i:} = \delta_i x^\top$. If Nora is applied to generate the update direction $d_{i:} = RN(\mathcal{P}_w^{r\perp}(g_{i:}))$, then as the network width $n \to \infty$, the inner product between the update direction and the input converges in probability to $\langle d_{i:}, x\rangle \xrightarrow{p} sgn(\delta_i)\sigma_x\sqrt{n}$. To achieve non-trivial feature learning such that $\Delta h_i = \Theta(1)$ as $n \to \infty$, the Nora learning rate must scale as $\eta = \frac{\eta_0}{\sqrt{n}}$.*

Proof. For output neuron i, Nora first removes the row-wise radial component of the gradient. Writing row vectors as columns inside inner products, this gives:

$$u_{i:} = \delta_i x - \frac{\delta_i\langle x, w_{i:}\rangle}{\|w_{i:}\|_2^2}w_{i:}. \tag{22}$$

We now compute the high-dimensional orders. Since x_j has variance σ_x^2 and $w_{i,j}$ has variance σ_w^2/n, the weak law of large numbers gives:

$$\|x\|_2^2 \xrightarrow{p} n\sigma_x^2 \implies \|x\|_2 = \sigma_x\sqrt{n} + o_p(\sqrt{n}). \tag{23}$$

Similarly:

$$\|w_{i:}\|_2^2 \xrightarrow{p} \sigma_w^2 = \Theta(1). \tag{24}$$

Because x and w are independent and centered, $\langle x, w_{i:}\rangle$ is a sum of n independent products with total variance $\sigma_x^2\sigma_w^2$. The central limit theorem gives:

$$\langle x, w_{i:}\rangle \xrightarrow{d} \mathcal{N}(0, \sigma_x^2\sigma_w^2) = \mathcal{O}_p(1). \tag{25}$$

Let $r_{i:}$ denote the radial component removed by the projection. Its Euclidean norm satisfies:

$$\|r_{i:}\|_2 = \left|\frac{\delta_i\langle x, w_{i:}\rangle}{\|w_{i:}\|_2^2}\right| \|w_{i:}\|_2 = \frac{|\delta_i||\mathcal{O}_p(1)|}{\sigma_w^2}\sigma_w = \mathcal{O}_p(1). \tag{26}$$

The primary gradient term $\delta_i x$ has norm $\Theta(\sqrt{n})$, while the removed radial component is $\mathcal{O}_p(1)$. Hence the projected vector satisfies:

$$\|u_{i:}\|_2 = \|\delta_i x - r_{i:}\|_2 = |\delta_i|\|x\|_2 + \mathcal{O}_p(1) = |\delta_i|\sigma_x\sqrt{n}(1 + o_p(1)). \tag{27}$$

Row-wise normalization gives:

$$d_{i:} = \frac{u_{i:}}{\|u_{i:}\|_2} = \frac{\delta_i x - r_{i:}}{|\delta_i|\sigma_x\sqrt{n}(1 + o_p(1))}. \tag{28}$$

The forward update depends on the inner product:

$$\langle d_{i:}, x\rangle = \frac{\delta_i\langle x, x\rangle - \langle r_{i:}, x\rangle}{|\delta_i|\sigma_x\sqrt{n}(1 + o_p(1))}. \tag{29}$$

Since $\langle x, x\rangle = n\sigma_x^2 + o_p(n)$ and $|\langle r_{i:}, x\rangle| \le \|r_{i:}\|_2\|x\|_2 = \mathcal{O}_p(\sqrt{n})$,

$$\langle d_{i:}, x\rangle = \frac{\delta_i n\sigma_x^2 - \mathcal{O}_p(\sqrt{n})}{|\delta_i|\sigma_x\sqrt{n}(1 + o_p(1))} = \operatorname{sgn}(\delta_i)\sigma_x\sqrt{n} + o_p(\sqrt{n}). \tag{30}$$

Thus $\langle d_{i:}, x\rangle \xrightarrow{p} \operatorname{sgn}(\delta_i)\sigma_x\sqrt{n}$ holds. The actual forward update is $\Delta h_i = -\eta\langle d_{i:}, x\rangle = \mp\eta\sigma_x\sqrt{n}$. To achieve $\Delta h_i = \Theta(1)$ for non-trivial feature learning, it is necessary that $\eta = \Theta(n^{-1/2})$.

A.3 Preliminaries for Theorem 4.7–Corollary 4.9

A.3.1 Setup

For the convergence analysis, let m denote the number of rows in $w \in \mathbb{R}^{m\times n}$. We reserve v_t for the momentum sequence. For any matrix w with nonzero rows, define the row-wise perpendicular projection:

$$[\mathcal{P}_w^{r\perp}(x)]_{i:} := x_{i:} - \frac{\langle x_{i:}, w_{i:}\rangle}{\|w_{i:}\|_2^2}\, w_{i:}. \tag{31}$$

This projection removes the row-wise radial component: $\langle[\mathcal{P}_w^{r\perp}(x)]_{i:}, w_{i:}\rangle = 0$ for every row i. Equivalently, a matrix z is row-wise perpendicular to w when $\langle z_{i:}, w_{i:}\rangle = 0$ for all i.

For $t = 0, \dots, T-1$, Nora is indexed as:

$$g_t = \nabla f(w_t; \xi_t), \qquad v_{t+1} = \beta v_t + (1-\beta)g_t, \qquad v_0 = 0, \tag{32}$$

$$v_{t+1}^{r\perp} := \mathcal{P}_{w_t}^{r\perp}(v_{t+1}), \qquad d_t = \mathrm{RN}\big(v_{t+1}^{r\perp}\big), \qquad w_{t+1} = w_t - \eta d_t. \tag{33}$$

Thus d_t is the row-wise normalization of the component of the momentum that is row-wise perpendicular to w_t. We use the convention $0/0 = 0$ in RN, applied row-wise:

$$[\mathrm{RN}(x)]_{i:} = \begin{cases} \dfrac{x_{i:}}{\|x_{i:}\|_2}, & x_{i:} \neq 0, \\ 0, & x_{i:} = 0. \end{cases} \tag{34}$$

For a matrix $x \in \mathbb{R}^{m\times n}$, define:

$$\|x\|_{1,2} := \sum_{i=1}^{m} \|x_{i:}\|_2, \qquad \|x\|_{\infty,2} := \max_{1\le i\le m} \|x_{i:}\|_2. \tag{35}$$

We also define the natural filtration:

$$\mathcal{F}_t := \sigma(w_0, \xi_0, \dots, \xi_{t-1}), \tag{36}$$

so that w_t is $\mathcal{F}_t$-measurable and g_t is sampled conditionally on $\mathcal{F}_t$. Finally, define the projected gradient and momentum-tracking errors:

$$\mathcal{G}_t := \mathcal{P}_{w_t}^{r\perp}\big(\nabla f(w_t)\big), \qquad e_t := v_{t+1} - \nabla f(w_t), \qquad \bar{e}_t := v_{t+1}^{r\perp} - \mathcal{G}_t = \mathcal{P}_{w_t}^{r\perp}(e_t). \tag{37}$$

The projected gradient $\mathcal{G}_t$ is the relevant stationarity measure because Nora uses directions that are row-wise perpendicular to w_t. Therefore radial components of $\nabla f(w_t)$ do not contribute to the descent inner product with d_t.

A.3.2 Auxiliary Lemmas

Lemma A.1 (Row-wise projection is non-expansive)**.** *For any $w \in \mathbb{R}^{m\times n}$ with nonzero rows and any $x \in \mathbb{R}^{m\times n}$,*

$$\|\mathcal{P}_w^{r\perp}(x)\|_F \le \|x\|_F, \qquad \|\mathcal{P}_w^{r\perp}(x)\|_{1,2} \le \|x\|_{1,2}, \qquad \|\mathcal{P}_w^{r\perp}(x)\|_{\infty,2} \le \|x\|_{\infty,2}. \tag{38}$$

Proof. For each row i, let

$$P_i := I_n - \frac{w_{i:}^\top w_{i:}}{\|w_{i:}\|_2^2}. \tag{39}$$

Then $[\mathcal{P}_w^{r\perp}(x)]_{i:} = x_{i:}P_i$. Since P_i is an orthogonal projector on $\mathbb{R}^n$,

$$\|x_{i:}P_i\|_2 \le \|x_{i:}\|_2. \tag{40}$$

Summing over rows gives the Frobenius and $\|\cdot\|_{1,2}$ bounds, while taking the maximum over rows gives the $\|\cdot\|_{\infty,2}$ bound.

Lemma A.2 (Basic geometry of Nora)**.** *Let $z = \mathcal{P}_w^{r\perp}(x)$ and $d = \mathrm{RN}(z)$. Then:*

1. $\langle d_{i:}, w_{i:}\rangle = 0$ *for every row i;*
2. $\|d\|_F \le \sqrt{m}$*;*
3. $\|d\|_{\infty,2} \le 1$*;*
4. $\langle z, d\rangle = \|z\|_{1,2}$*;*
5. $\langle z, d\rangle \ge \|z\|_F$.

Proof. Each row $z_{i:}$ is orthogonal to $w_{i:}$. Every nonzero row of d is a scalar multiple of $z_{i:}$, so it is also orthogonal to $w_{i:}$. Thus d is row-wise perpendicular to w.

For every row, $\|d_{i:}\|_2 \le 1$. Thus:

$$\|d\|_F^2 = \sum_{i=1}^m \|d_{i:}\|_2^2 \le m, \qquad \|d\|_{\infty,2} = \max_i \|d_{i:}\|_2 \le 1. \tag{41}$$

Moreover,

$$\langle z, d\rangle = \sum_{i=1}^m \left\langle z_{i:}, \frac{z_{i:}}{\|z_{i:}\|_2}\right\rangle = \sum_{i=1}^m \|z_{i:}\|_2 = \|z\|_{1,2}. \tag{42}$$

Finally,

$$\|z\|_{1,2} = \sum_i \|z_{i:}\|_2 \ge \left(\sum_i \|z_{i:}\|_2^2\right)^{1/2} = \|z\|_F, \tag{43}$$

which gives $\langle z, d\rangle \ge \|z\|_F$.

Lemma A.3 (Descent under Frobenius smoothness)**.** *Under Assumption 4.3(a), for every t,*

$$f(w_t) - f(w_{t+1}) \ge \eta\langle \nabla f(w_t), d_t\rangle - \frac{L_F\eta^2 m}{2}. \tag{44}$$

Proof. Let $\Delta_t := w_{t+1} - w_t$. By the fundamental theorem of calculus,

$$f(w_t + \Delta_t) - f(w_t) = \int_0^1 \langle \nabla f(w_t + s\Delta_t), \Delta_t\rangle\, ds. \tag{45}$$

Subtracting $\langle \nabla f(w_t), \Delta_t\rangle$ and applying Cauchy–Schwarz gives:

$$f(w_t + \Delta_t) - f(w_t) - \langle \nabla f(w_t), \Delta_t\rangle = \int_0^1 \langle \nabla f(w_t + s\Delta_t) - \nabla f(w_t), \Delta_t\rangle\, ds \tag{46}$$

$$\le \int_0^1 \|\nabla f(w_t + s\Delta_t) - \nabla f(w_t)\|_F \,\|\Delta_t\|_F\, ds. \tag{47}$$

Assumption 4.3(a) implies:

$$\|\nabla f(w_t + s\Delta_t) - \nabla f(w_t)\|_F \le L_F s \|\Delta_t\|_F. \tag{48}$$

Hence the standard quadratic upper bound follows from the stated smoothness assumption:

$$f(w_t + \Delta_t) \le f(w_t) + \langle \nabla f(w_t), \Delta_t \rangle + \frac{L_F}{2}\|\Delta_t\|_F^2. \tag{49}$$

Substituting $\Delta_t = -\eta d_t$ gives:

$$f(w_{t+1}) \le f(w_t) - \eta \langle \nabla f(w_t), d_t \rangle + \frac{L_F \eta^2}{2}\|d_t\|_F^2. \tag{50}$$

Using Lemma A.2(ii), $\|d_t\|_F^2 \le m$, which yields the claim.

Lemma A.4 (Descent under matched $(\infty, 2)$-smoothness)**.** *Under Assumption 4.3(b), for every t,*

$$f(w_t) - f(w_{t+1}) \ge \eta \langle \nabla f(w_t), d_t \rangle - \frac{L_{\infty,2}\eta^2}{2}. \tag{51}$$

Proof. Let $\Delta_t := w_{t+1} - w_t = -\eta d_t$. By the fundamental theorem of calculus,

$$f(w_t + \Delta_t) - f(w_t) = \langle \nabla f(w_t), \Delta_t \rangle + \int_0^1 \langle \nabla f(w_t + s\Delta_t) - \nabla f(w_t), \Delta_t \rangle \, ds. \tag{52}$$

For any matrices A and B, the $(1,2)$–$(\infty,2)$ duality bound gives:

$$|\langle A, B \rangle| \le \|A\|_{1,2} \|B\|_{\infty,2}. \tag{53}$$

Therefore, by Assumption 4.3(b),

$$|\langle \nabla f(w_t + s\Delta_t) - \nabla f(w_t), \Delta_t \rangle| \le \|\nabla f(w_t + s\Delta_t) - \nabla f(w_t)\|_{1,2} \|\Delta_t\|_{\infty,2} \tag{54}$$

$$\le L_{\infty,2} s \|\Delta_t\|_{\infty,2}^2. \tag{55}$$

Lemma A.2(iii) gives $\|\Delta_t\|_{\infty,2} = \eta \|d_t\|_{\infty,2} \le \eta$. Hence:

$$f(w_{t+1}) - f(w_t) \le -\eta \langle \nabla f(w_t), d_t \rangle + \int_0^1 L_{\infty,2} s \eta^2 \, ds = -\eta \langle \nabla f(w_t), d_t \rangle + \frac{L_{\infty,2}\eta^2}{2}. \tag{56}$$

Rearranging proves the claim.

Lemma A.5 (Projected inner-product lower bound in Frobenius norm)**.** *For every t,*

$$\langle \nabla f(w_t), d_t \rangle \ge \|\mathcal{G}_t\|_F - (\sqrt{m} + 1)\|e_t\|_F. \tag{57}$$

Proof. Since d_t is row-wise perpendicular to w_t by Lemma A.2(i), and $\nabla f(w_t) - \mathcal{G}_t$ is row-wise parallel to w_t, we have

$$\langle \nabla f(w_t), d_t \rangle = \langle \mathcal{G}_t, d_t \rangle. \tag{58}$$

Since $\mathcal{G}_t = v_{t+1}^{r\perp} - \bar{e}_t$,

$$\langle \mathcal{G}_t, d_t \rangle = \langle v_{t+1}^{r\perp}, d_t \rangle - \langle \bar{e}_t, d_t \rangle. \tag{59}$$

By Lemma A.2(v),

$$\langle v_{t+1}^{r\perp}, d_t \rangle \ge \|v_{t+1}^{r\perp}\|_F. \tag{60}$$

By Cauchy–Schwarz and Lemma A.2(ii),

$$|\langle \bar{e}_t, d_t \rangle| \le \|\bar{e}_t\|_F \|d_t\|_F \le \sqrt{m}\, \|\bar{e}_t\|_F. \tag{61}$$

Also,

$$\|v_{t+1}^{r\perp}\|_F = \|\mathcal{G}_t + \bar{e}_t\|_F \ge \|\mathcal{G}_t\|_F - \|\bar{e}_t\|_F. \tag{62}$$

Combining the above,

$$\langle \nabla f(w_t), d_t \rangle \ge \|\mathcal{G}_t\|_F - (\sqrt{m} + 1)\|\bar{e}_t\|_F. \tag{63}$$

Finally, by Lemma A.1,

$$\|\bar{e}_t\|_F = \|\mathcal{P}_{w_t}^{r\perp}(e_t)\|_F \le \|e_t\|_F. \tag{64}$$

This proves the claim.

Lemma A.6 (Projected inner-product lower bound in $(1,2)$-norm)**.** *For every t,*

$$\langle \nabla f(w_t), d_t \rangle \geq \|\mathcal{G}_t\|_{1,2} - 2\|e_t\|_{1,2}. \tag{65}$$

Proof. As in Lemma A.5,

$$\langle \nabla f(w_t), d_t \rangle = \langle \mathcal{G}_t, d_t \rangle = \langle v_{t+1}^{r\perp}, d_t \rangle - \langle \bar{e}_t, d_t \rangle. \tag{66}$$

By Lemma A.2(iv),

$$\langle v_{t+1}^{r\perp}, d_t \rangle = \|v_{t+1}^{r\perp}\|_{1,2}. \tag{67}$$

By duality and Lemma A.2(iii),

$$|\langle \bar{e}_t, d_t \rangle| \leq \|\bar{e}_t\|_{1,2} \|d_t\|_{\infty,2} \leq \|\bar{e}_t\|_{1,2}. \tag{68}$$

Also,

$$\|v_{t+1}^{r\perp}\|_{1,2} = \|\mathcal{G}_t + \bar{e}_t\|_{1,2} \geq \|\mathcal{G}_t\|_{1,2} - \|\bar{e}_t\|_{1,2}. \tag{69}$$

Hence,

$$\langle \nabla f(w_t), d_t \rangle \geq \|\mathcal{G}_t\|_{1,2} - 2\|\bar{e}_t\|_{1,2}. \tag{70}$$

Using Lemma A.1,

$$\|\bar{e}_t\|_{1,2} \leq \|e_t\|_{1,2}, \tag{71}$$

and the proof is complete.

Assumption A.7 (Unbiased stochastic gradients)**.** For all t,

$$\mathbb{E}[g_t \mid \mathcal{F}_t] = \nabla f(w_t). \tag{72}$$

Assumption A.8 (Bounded variance)**.** There exists $\sigma > 0$ such that, for all t,

$$\mathbb{E}\big[\|g_t - \nabla f(w_t)\|_F^2 \mid \mathcal{F}_t\big] \leq \frac{\sigma^2}{B}, \tag{73}$$

where B is the batch size.

Assumption A.9 (Lower bounded objective)**.** The objective is bounded below by $f^\star$. We write $\Delta := f(w_0) - f^\star$.

Lemma A.10 (Momentum tracking under Frobenius smoothness)**.** *Under Assumptions 4.3(a), A.7, and A.8,*

$$\sum_{t=0}^{T-1} \mathbb{E}\big[\|e_t\|_F\big] \leq (T-1)\frac{L_F \eta \sqrt{m}\beta}{1-\beta} + T\frac{\sigma}{\sqrt{B}}\sqrt{\frac{1-\beta}{1+\beta}}. \tag{74}$$

Proof. Let $\zeta_t := g_t - \nabla f(w_t)$. Then

$$\mathbb{E}[\zeta_t \mid \mathcal{F}_t] = 0 \tag{75}$$

by Assumption A.7, and

$$\mathbb{E}\big[\|\zeta_t\|_F^2 \mid \mathcal{F}_t\big] \leq \frac{\sigma^2}{B} \tag{76}$$

by Assumption A.8.

For $t \geq 1$,

$$\begin{aligned} e_t &= v_{t+1} - \nabla f(w_t) \\ &= \beta v_t + (1-\beta) g_t - \nabla f(w_t) \\ &= \beta\big(v_t - \nabla f(w_{t-1})\big) + \beta\big(\nabla f(w_{t-1}) - \nabla f(w_t)\big) + (1-\beta)\zeta_t \\ &= \beta e_{t-1} + \beta\big(\nabla f(w_{t-1}) - \nabla f(w_t)\big) + (1-\beta)\zeta_t, \end{aligned} \tag{77}$$

while $e_0 = (1-\beta)\zeta_0$. Unrolling the recursion gives

$$e_t = \sum_{j=0}^{t} \beta^{t-j}(1-\beta)\zeta_j + \sum_{j=1}^{t} \beta^{t-j+1}\big(\nabla f(w_{j-1}) - \nabla f(w_j)\big). \tag{78}$$

Therefore,

$$\|e_t\|_F \le \left\|\sum_{j=0}^{t} \beta^{t-j}(1-\beta)\zeta_j\right\|_F + \sum_{j=1}^{t} \beta^{t-j+1}\|\nabla f(w_{j-1}) - \nabla f(w_j)\|_F. \tag{79}$$

Using Assumption 4.3(a), $w_j - w_{j-1} = -\eta d_{j-1}$, and Lemma A.2(ii),

$$\begin{aligned}\sum_{j=1}^{t} \beta^{t-j+1}\|\nabla f(w_{j-1}) - \nabla f(w_j)\|_F &\le \sum_{j=1}^{t} \beta^{t-j+1} L_F \|w_{j-1} - w_j\|_F \\ &= \sum_{j=1}^{t} \beta^{t-j+1} L_F \eta \|d_{j-1}\|_F \\ &\le L_F \eta \sqrt{m} \sum_{k=1}^{t} \beta^k \\ &\le L_F \eta \sqrt{m} \frac{\beta}{1-\beta}.\end{aligned} \tag{80}$$

For the noise term, Jensen's inequality implies

$$\mathbb{E}\left\|\sum_{j=0}^{t} \beta^{t-j}(1-\beta)\zeta_j\right\|_F \le \sqrt{\mathbb{E}\left\|\sum_{j=0}^{t} \beta^{t-j}(1-\beta)\zeta_j\right\|_F^2}. \tag{81}$$

Since $\{\zeta_t\}$ is a martingale difference sequence, the cross terms vanish, and thus

$$\begin{aligned}\mathbb{E}\left\|\sum_{j=0}^{t} \beta^{t-j}(1-\beta)\zeta_j\right\|_F^2 &= \sum_{j=0}^{t} \beta^{2(t-j)}(1-\beta)^2 \mathbb{E}\|\zeta_j\|_F^2 \\ &\le \frac{\sigma^2}{B}(1-\beta)^2 \sum_{k=0}^{t} \beta^{2k} \\ &\le \frac{\sigma^2}{B}\frac{1-\beta}{1+\beta}.\end{aligned} \tag{82}$$

Hence,

$$\mathbb{E}\left\|\sum_{j=0}^{t} \beta^{t-j}(1-\beta)\zeta_j\right\|_F \le \frac{\sigma}{\sqrt{B}}\sqrt{\frac{1-\beta}{1+\beta}}. \tag{83}$$

Combining the previous bounds and summing over $t = 0, \dots, T-1$ yields

$$\sum_{t=0}^{T-1} \mathbb{E}\big[\|e_t\|_F\big] \le (T-1)\frac{L_F \eta \sqrt{m}\beta}{1-\beta} + T\frac{\sigma}{\sqrt{B}}\sqrt{\frac{1-\beta}{1+\beta}}. \tag{84}$$

Lemma A.11 (Momentum tracking under matched $(\infty, 2)$-smoothness)**.** *Under Assumptions 4.3(b), A.7, and A.8,*

$$\sum_{t=0}^{T-1} \mathbb{E}\big[\|e_t\|_{1,2}\big] \le (T-1)\frac{L_{\infty,2}\eta\beta}{1-\beta} + T\frac{\sqrt{m}\sigma}{\sqrt{B}}\sqrt{\frac{1-\beta}{1+\beta}}. \tag{85}$$

Proof. The recursion for e_t is the same as in Lemma A.10. Hence,

$$e_t = \sum_{j=0}^{t} \beta^{t-j}(1-\beta)\zeta_j + \sum_{j=1}^{t} \beta^{t-j+1}\big(\nabla f(w_{j-1}) - \nabla f(w_j)\big). \tag{86}$$

By the triangle inequality,

$$\|e_t\|_{1,2} \le \left\| \sum_{j=0}^{t} \beta^{t-j}(1-\beta)\zeta_j \right\|_{1,2} + \sum_{j=1}^{t} \beta^{t-j+1} \|\nabla f(w_{j-1}) - \nabla f(w_j)\|_{1,2}. \tag{87}$$

Using Assumption 4.3(b), $w_j - w_{j-1} = -\eta d_{j-1}$, and Lemma A.2(iii),

$$\begin{aligned} \sum_{j=1}^{t} \beta^{t-j+1} \|\nabla f(w_{j-1}) - \nabla f(w_j)\|_{1,2} &\le \sum_{j=1}^{t} \beta^{t-j+1} L_{\infty,2} \|w_{j-1} - w_j\|_{\infty,2} \\ &= \sum_{j=1}^{t} \beta^{t-j+1} L_{\infty,2} \eta \|d_{j-1}\|_{\infty,2} \\ &\le L_{\infty,2}\eta \sum_{k=1}^{t} \beta^{k} \\ &\le L_{\infty,2}\eta \frac{\beta}{1-\beta}. \end{aligned} \tag{88}$$

For the noise term, we use $\|A\|_{1,2} \le \sqrt{m}\|A\|_F$, hence

$$\mathbb{E}\left\| \sum_{j=0}^{t} \beta^{t-j}(1-\beta)\zeta_j \right\|_{1,2} \le \sqrt{m}\, \mathbb{E}\left\| \sum_{j=0}^{t} \beta^{t-j}(1-\beta)\zeta_j \right\|_{F}. \tag{89}$$

Applying the Frobenius-noise bound from the proof of Lemma A.10,

$$\mathbb{E}\left\| \sum_{j=0}^{t} \beta^{t-j}(1-\beta)\zeta_j \right\|_{1,2} \le \frac{\sqrt{m}\sigma}{\sqrt{B}} \sqrt{\frac{1-\beta}{1+\beta}}. \tag{90}$$

Summing over $t = 0, \dots, T-1$ proves the lemma.

A.4 Proof of Theorem 4.7

Theorem 4.7. *Suppose Assumptions 4.3(b), A.7, A.8, and A.9 hold, and Nora uses a constant step size $\eta_t = \eta$. Then:*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\big[\|\mathcal{G}_t\|_{1,2}\big] \le \frac{\Delta}{T\eta} + 2\left[\left(1 - \frac{1}{T}\right)\frac{L_{\infty,2}\eta\beta}{1-\beta} + \frac{\sqrt{m}\sigma}{\sqrt{B}}\sqrt{\frac{1-\beta}{1+\beta}}\right] + \frac{L_{\infty,2}\eta}{2}. \tag{91}$$

With the parameter choice:

$$\eta = \sqrt{\frac{(1-\beta)\Delta}{L_{\infty,2}T}}, \qquad 1-\beta = \min\left\{\frac{\sqrt{L_{\infty,2}\Delta}}{2\sqrt{m}\sigma\sqrt{T}},\, 1\right\}, \tag{92}$$

Nora reaches an ϵ-stationary point in the projected $\|\cdot\|_{1,2}$ sense with complexity:

$$T = \mathcal{O}\big(mL_{\infty,2}\sigma^2\Delta\,\epsilon^{-4}\big). \tag{93}$$

Proof. Summing Lemma A.4 over $t = 0, \dots, T-1$, we obtain

$$f(w_0) - f(w_T) \ge \eta \sum_{t=0}^{T-1} \langle \nabla f(w_t), d_t\rangle - \frac{TL_{\infty,2}\eta^2}{2}. \tag{94}$$

By Assumption A.9,

$$\Delta \ge \eta \sum_{t=0}^{T-1} \langle \nabla f(w_t), d_t\rangle - \frac{TL_{\infty,2}\eta^2}{2}. \tag{95}$$

Applying Lemma A.6,

$$\Delta \geq \eta \sum_{t=0}^{T-1} \|\mathcal{G}_t\|_{1,2} - 2\eta \sum_{t=0}^{T-1} \|e_t\|_{1,2} - \frac{TL_{\infty,2}\eta^2}{2}. \tag{96}$$

Taking expectations and using Lemma A.11,

$$\begin{aligned} \eta \sum_{t=0}^{T-1} \mathbb{E}\big[\|\mathcal{G}_t\|_{1,2}\big] &\leq \Delta + 2\eta \sum_{t=0}^{T-1} \mathbb{E}\big[\|e_t\|_{1,2}\big] + \frac{TL_{\infty,2}\eta^2}{2} \\ &\leq \Delta + 2\eta \left[(T-1)\frac{L_{\infty,2}\eta\beta}{1-\beta} + T\frac{\sqrt{m}\sigma}{\sqrt{B}}\sqrt{\frac{1-\beta}{1+\beta}}\right] + \frac{TL_{\infty,2}\eta^2}{2}. \end{aligned} \tag{97}$$

Dividing both sides by $T\eta$ yields

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\big[\|\mathcal{G}_t\|_{1,2}\big] \leq \frac{\Delta}{T\eta} + 2\left[\left(1-\frac{1}{T}\right)\frac{L_{\infty,2}\eta\beta}{1-\beta} + \frac{\sqrt{m}\sigma}{\sqrt{B}}\sqrt{\frac{1-\beta}{1+\beta}}\right] + \frac{L_{\infty,2}\eta}{2}. \tag{98}$$

This proves the explicit bound.

To derive the rate, set $q := 1-\beta$. For $B = 1$, the preceding bound implies, up to universal constants,

$$R_T \lesssim \frac{\Delta}{T\eta} + \frac{L_{\infty,2}\eta}{q} + \sqrt{m}\sigma\sqrt{q} + L_{\infty,2}\eta. \tag{99}$$

Since $q \leq 1$, the last term is dominated by $L_{\infty,2}\eta/q$. For a fixed q, we balance the descent term and the momentum-tracking term:

$$\frac{\Delta}{T\eta} \asymp \frac{L_{\infty,2}\eta}{q}, \qquad \text{which gives} \qquad \eta = \sqrt{\frac{q\Delta}{L_{\infty,2}T}}. \tag{100}$$

With this choice, the bound reduces to

$$R_T \lesssim \sqrt{\frac{L_{\infty,2}\Delta}{qT}} + \sqrt{m}\sigma\sqrt{q}. \tag{101}$$

The first term increases as q decreases, whereas the stochastic-noise term decreases. Balancing them gives $q \asymp \sqrt{L_{\infty,2}\Delta}/(\sqrt{m}\sigma\sqrt{T})$. Thus we take

$$q = 1-\beta = \min\left\{\frac{\sqrt{L_{\infty,2}\Delta}}{2\sqrt{m}\sigma\sqrt{T}}, 1\right\}, \qquad \eta = \sqrt{\frac{(1-\beta)\Delta}{L_{\infty,2}T}}. \tag{102}$$

This is a constructive parameter choice used to obtain the complexity bound; it is not meant to be a unique tuning rule for practical training.

A.5 Proof of Proposition 4.8

Proposition 4.8. *Suppose Assumptions 4.3(a), A.7, A.8, and A.9 hold, and Nora uses a constant step size $\eta_t = \eta$. Then:*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\big[\|\mathcal{G}_t\|_F\big] \leq \frac{\Delta}{T\eta} + (\sqrt{m}+1)\left[\left(1-\frac{1}{T}\right)\frac{L_F\eta\sqrt{m}\beta}{1-\beta} + \frac{\sigma}{\sqrt{B}}\sqrt{\frac{1-\beta}{1+\beta}}\right] + \frac{L_F\eta m}{2}, \tag{103}$$

and similarly,

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\big[\|\mathcal{G}_t\|_{1,2}\big] \leq \frac{\Delta}{T\eta} + 2\left[\left(1-\frac{1}{T}\right)\frac{L_F\eta m\beta}{1-\beta} + \frac{\sqrt{m}\sigma}{\sqrt{B}}\sqrt{\frac{1-\beta}{1+\beta}}\right] + \frac{L_F\eta m}{2}. \tag{104}$$

Consequently, Nora reaches an ϵ-stationary point in either projected measure with complexity:

$$T = \mathcal{O}\big(m^2 L_F \sigma^2 \Delta\, \epsilon^{-4}\big). \tag{105}$$

Proof. We first prove the Frobenius-norm bound. Summing Lemma A.3 over $t = 0, \dots, T-1$ gives

$$f(w_0) - f(w_T) \geq \eta \sum_{t=0}^{T-1} \langle \nabla f(w_t), d_t \rangle - \frac{TL_F \eta^2 m}{2}. \tag{106}$$

By Assumption A.9, $f(w_T) \geq f^\star$, hence

$$\Delta \geq \eta \sum_{t=0}^{T-1} \langle \nabla f(w_t), d_t \rangle - \frac{TL_F \eta^2 m}{2}. \tag{107}$$

Applying Lemma A.5,

$$\Delta \geq \eta \sum_{t=0}^{T-1} \|\mathcal{G}_t\|_F - \eta(\sqrt{m}+1) \sum_{t=0}^{T-1} \|e_t\|_F - \frac{TL_F \eta^2 m}{2}. \tag{108}$$

Taking expectation and using Lemma A.10,

$$\begin{aligned} \eta \sum_{t=0}^{T-1} \mathbb{E}\big[\|\mathcal{G}_t\|_F\big] &\leq \Delta + \eta(\sqrt{m}+1) \sum_{t=0}^{T-1} \mathbb{E}\big[\|e_t\|_F\big] + \frac{TL_F \eta^2 m}{2} \\ &\leq \Delta + \eta(\sqrt{m}+1) \left[(T-1) \frac{L_F \eta \sqrt{m} \beta}{1-\beta} + T \frac{\sigma}{\sqrt{B}} \sqrt{\frac{1-\beta}{1+\beta}} \right] + \frac{TL_F \eta^2 m}{2}. \end{aligned} \tag{109}$$

Dividing both sides by $T\eta$ yields

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\big[\|\mathcal{G}_t\|_F\big] \leq \frac{\Delta}{T\eta} + (\sqrt{m}+1) \left[\left(1 - \frac{1}{T}\right) \frac{L_F \eta \sqrt{m} \beta}{1-\beta} + \frac{\sigma}{\sqrt{B}} \sqrt{\frac{1-\beta}{1+\beta}} \right] + \frac{L_F \eta m}{2}. \tag{110}$$

We next prove the $\|\cdot\|_{1,2}$ bound under the same Frobenius smoothness assumption. Starting again from Lemma A.3,

$$\Delta \geq \eta \sum_{t=0}^{T-1} \langle \nabla f(w_t), d_t \rangle - \frac{TL_F \eta^2 m}{2}. \tag{111}$$

Using Lemma A.6,

$$\Delta \geq \eta \sum_{t=0}^{T-1} \|\mathcal{G}_t\|_{1,2} - 2\eta \sum_{t=0}^{T-1} \|e_t\|_{1,2} - \frac{TL_F \eta^2 m}{2}. \tag{112}$$

Since $\|A\|_{1,2} \leq \sqrt{m}\|A\|_F$, Lemma A.10 implies

$$\sum_{t=0}^{T-1} \mathbb{E}\big[\|e_t\|_{1,2}\big] \leq \sqrt{m} \sum_{t=0}^{T-1} \mathbb{E}\big[\|e_t\|_F\big] \leq (T-1) \frac{L_F \eta m \beta}{1-\beta} + T \frac{\sqrt{m}\sigma}{\sqrt{B}} \sqrt{\frac{1-\beta}{1+\beta}}. \tag{113}$$

Substituting the above bound and dividing by $T\eta$ gives

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\big[\|\mathcal{G}_t\|_{1,2}\big] \leq \frac{\Delta}{T\eta} + 2 \left[\left(1 - \frac{1}{T}\right) \frac{L_F \eta m \beta}{1-\beta} + \frac{\sqrt{m}\sigma}{\sqrt{B}} \sqrt{\frac{1-\beta}{1+\beta}} \right] + \frac{L_F \eta m}{2}. \tag{114}$$

It remains to justify the stated complexity. Take $B = 1$ and choose:

$$\eta = \sqrt{\frac{(1-\beta)\Delta}{L_F m T}}, \qquad 1 - \beta = \min\left\{ \frac{\sqrt{L_F \Delta}}{2\sigma\sqrt{T}}, 1 \right\}. \tag{115}$$

Substituting this choice into either of the two preceding bounds gives

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\big[\|\mathcal{G}_t\|_F\big] = \mathcal{O}\left(\sqrt[4]{\frac{m^2 L_F \sigma^2 \Delta}{T}} + \sqrt{\frac{m L_F \Delta}{T}} \right), \tag{116}$$

and similarly,

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[\|\mathcal{G}_t\|_{1,2}\big] = \mathcal{O}\left(\sqrt[4]{\frac{m^2 L_F \sigma^2 \Delta}{T}} + \sqrt{\frac{m L_F \Delta}{T}}\right). \tag{117}$$

Thus Nora reaches an ϵ-stationary point in either projected measure with iteration complexity:

$$T = \mathcal{O}\big(m^2 L_F \sigma^2 \Delta\, \epsilon^{-4}\big). \tag{118}$$

A.6 Proof of Corollary 4.9

Corollary 4.9. *Assume, in addition, that f is row-wise scale invariant, namely:*

$$f(Dw) = f(w), \qquad \forall w \in \mathbb{R}^{m\times n},\ \forall D \succ 0 \text{ diagonal}. \tag{119}$$

Then $\nabla f(w_t)$ is row-wise perpendicular to w_t for all t, i.e.,

$$\mathcal{P}_{w_t}^{r\perp}\big(\nabla f(w_t)\big) = \nabla f(w_t). \tag{120}$$

Therefore, Theorem 4.4 and Proposition 4.5 hold with $\mathcal{G}_t$ replaced by $\nabla f(w_t)$.

Proof. Fix any row index i, and define the diagonal matrix

$$D_i(c) := I + (c-1)e_i e_i^\top, \qquad c > 0. \tag{121}$$

By row-wise scale invariance,

$$f(D_i(c)w) = f(w), \qquad \forall c > 0. \tag{122}$$

Differentiating both sides with respect to c at $c = 1$, we obtain

$$0 = \frac{d}{dc} f(D_i(c)w)\bigg|_{c=1} = \langle \nabla f(w)_{i:},\, w_{i:}\rangle. \tag{123}$$

Since i is arbitrary, every row of $\nabla f(w)$ is orthogonal to the corresponding row of w. Hence

$$\mathcal{P}_{w}^{r\perp}\big(\nabla f(w)\big) = \nabla f(w). \tag{124}$$

Applying this identity to every iterate w_t shows that

$$\mathcal{G}_t = \nabla f(w_t), \qquad \forall t. \tag{125}$$

Therefore, the statements of Theorem 4.4 and Proposition 4.5 hold with $\mathcal{G}_t$ replaced by $\nabla f(w_t)$.

Table 5: Detailed model and training configurations.

Configuration	60M	135M
Layers	8	12
Hidden size	512	768
Attention heads	8	12
MLP size	1376	2048
Maximum context length	256	256
Training steps	10,000	20,000
GPUs	2	4
Per-GPU batch size	64	64
Gradient accumulation	4	2
Global batch size	512	512
Nominal token budget	1.31B	2.62B
Base learning rate	1×10^{-3}	1×10^{-3}
Auxiliary Adam learning rate	5×10^{-3}	3×10^{-3}
Warmup steps	1,000	2,000
Learning-rate schedule	cosine	cosine
Gradient clipping	1.0	1.0
Precision	bf16	bf16
Evaluation frequency	every 1,000 steps	every 1,000 steps
Checkpoint frequency	every 5,000 steps	every 5,000 steps

Table 6: Matrix learning-rate sweep grids.

Model size	Optimizer	Weight decay	Matrix learning-rate grid
60M	Muon	0.1	$\{0.005, 0.01, 0.02, 0.03, 0.04\}$
60M	Mano	0.1	$\{0.001, 0.004, 0.005, 0.01, 0.02\}$
60M	RMNP	0.1	$\{0.001, 0.004, 0.005, 0.01, 0.02\}$
60M	Nora	0.0	$\{0.001, 0.004, 0.005, 0.01, 0.02\}$
135M	Muon	0.1	$\{0.005, 0.01, 0.02, 0.03\}$
135M	Mano	0.1	$\{0.003, 0.005, 0.01, 0.02\}$
135M	RMNP	0.1	$\{0.003, 0.005, 0.01, 0.02\}$
135M	Nora	0.0	$\{0.003, 0.005, 0.01, 0.02\}$

B Additional Experimental Details

B.1 Model and Training Configurations

Table 5 reports the detailed model and training configurations used in the 60M and 135M experiments. Both settings use context length 256, global batch size 512, cosine learning-rate decay, bf16 precision, gradient clipping at 1.0, evaluation every 1,000 steps, and checkpointing every 5,000 steps.

B.2 Matrix Learning-rate Sweep Grids

Table 6 lists the matrix learning-rate sweep grids for all optimizers. Nora uses the same matrix learning-rate grid as Mano and RMNP, but differs in weight decay: Nora uses weight decay 0, while Muon, Mano, and RMNP use weight decay 0.1.

B.3 Other Experimental Results

In Tables 7, 8, 9, and 10, we present the perplexity and loss results across 60M and 135M model scales, evaluated under various algorithms and learning rate configurations. As illustrated, Nora demonstrates a significant advantage over all compared baselines across these diverse settings.

Table 7: Perplexity results of all baselines under different learning rates. (60M)

Perplexity	0.001	0.004	0.01	0.02
Mano	34.09	29.55	31.53	34.39
RMNP	35.42	30.12	30.30	93.26
Muon	41.65	31.44	31.09	31.99
Nora	31.24	28.94	29.89	31.17

Table 8: Loss results of all baselines under different learning rates. (60M)

Loss	0.001	0.004	0.01	0.02
Mano	3.529	3.386	3.451	3.538
RMNP	3.567	3.405	3.411	4.535
Muon	3.729	3.448	3.437	3.466
Nora	3.442	3.365	3.398	3.440

Table 9: Perplexity results of all baselines under different learning rates. (135M)

Matrix LR	0.003	0.005	0.01	0.02
Mano	22.13	22.14	23.34	25.07
Nora	21.74	21.86	22.43	23.39
Matrix LR	0.005	0.01	0.02	0.03
RMNP	22.62	22.46	22.64	22.67
Muon	23.40	23.17	24.06	23.23

Table 10: Loss results of all baselines under different learning rates. (135M)

Matrix LR	0.003	0.005	0.01	0.02
Mano	3.097	3.098	3.150	3.222
Nora	3.079	3.085	3.111	3.152
Matrix LR	0.005	0.01	0.02	0.03
RMNP	3.119	3.112	3.120	3.121
Muon	3.153	3.143	3.180	3.146

C Running and Reference Code

C.1 Run Quickly

We provide a dedicated repository containing the Nora[3] source code, along with detailed rationales and instructions for replacing Adam with Nora. Additionally, a separate reproduction repository[4] is made available to facilitate the rapid replication of all experimental results presented in this paper.

Please set up the C4 [38] dataset yourself and download the reproduction repository, then run:

```
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"

exec "$SCRIPT_DIR/train_universal.sh" \
    --model_size 135m \
    --optimizer nora \
    --num_gpus 4 \
    --lr_matrix 0.005 \
    --lr_adam 0.02 \
    --num_steps 20000 \
    --batch_size 64 \
    --total_batch_size 512 \
    --warmup_steps 2000 \
    --weight_decay 0.1 \
    --save_every 10000 \
    --eval_every 1000 \
    "$@"
```

It should be noted that the results for Muon, RMNP, and Mano can be reproduced using the exact same experimental setup as Nora.

C.2 Reference Code

```
import math

import torch
import torch.nn.functional as F


LOW_PRECISION_DTYPES = (torch.float16, torch.bfloat16)


class Nora(torch.optim.Optimizer):
    """Normalized Orthogonal Row Alignment optimizer for scalable
       matrix training."""

    def __init__(
        self,
        param_groups,
        lr_nora=0.005,
        lr_adam=0.001,
        momentum=0.95,
        beta=0.95,
        weight_decay=0.0,
        betas=(0.9, 0.95),
        eps=1e-10,
    ):
        defaults = dict(
            lr_nora=lr_nora,
            lr_adam=lr_adam,
            momentum=momentum,
            beta=beta,
```

[3] https://github.com/Yuan-Jinghui/Nora
[4] https://github.com/JiaxuanZou0714/Lrp

```
            weight_decay=weight_decay,
            betas=betas,
            eps=eps,
        )
        super().__init__(param_groups, defaults)

    def step(self, closure=None):
        loss = None
        if closure is not None:
            loss = closure()

        for group in self.param_groups:
            lr = group["lr"]
            momentum = group.get("momentum", 0.95)
            beta = group.get("beta", 0.95)
            weight_decay = group.get("weight_decay", 0.0)
            betas = group.get("betas", (0.9, 0.95))
            eps = group.get("eps", 1e-10)
            is_nora = group.get("is_nora", True)

            for p in group["params"]:
                if p.grad is None:
                    continue

                grad = p.grad.data
                param_state = self.state.setdefault(p, {})

                use_master_param = p.data.dtype in
                    LOW_PRECISION_DTYPES
                if use_master_param:
                    if "fp32_param" not in param_state:
                        param_state["fp32_param"] = p.data.detach().
                            float().clone()
                    elif param_state["fp32_param"].dtype != torch.
                        float32:
                        param_state["fp32_param"] = param_state["
                            fp32_param"].float()
                    param_data = param_state["fp32_param"]
                    grad_data = grad.float()
                else:
                    param_data = p.data
                    grad_data = grad

                if is_nora and grad.dim() >= 2:
                    if "momentum_buffer" not in param_state:
                        buf = torch.zeros_like(grad_data)
                    else:
                        buf = param_state["momentum_buffer"]
                        if use_master_param and buf.dtype != torch.
                            float32:
                            buf = buf.float()

                    buf.lerp_(grad_data, 1 - beta)
                    m_t = grad_data.lerp(buf, momentum)

                    theta_hat = F.normalize(param_data, p=2, dim=-1,
                        eps=eps)

                    dot_product = torch.sum(m_t * theta_hat, dim=-1,
                        keepdim=True)
                    v = m_t - dot_product * theta_hat

                    v_hat = F.normalize(v, p=2, dim=-1, eps=eps)
```

```
                    scale = max(1, math.sqrt(grad_data.size(-2) /
                        grad_data.size(-1)))
                    update_direction = v_hat * scale

                    if weight_decay > 0:
                        param_data.mul_(1 - lr * weight_decay)

                    param_data.add_(update_direction, alpha=-lr)

                    if use_master_param:
                        p.data.copy_(param_data.to(dtype=p.data.dtype)
                            )

                    param_state["momentum_buffer"] = buf

                else:
                    if "exp_avg" not in param_state:
                        param_state["exp_avg"] = torch.zeros_like(
                            grad_data)
                        param_state["exp_avg_sq"] = torch.zeros_like(
                            grad_data)
                        param_state["step"] = 0
                    elif use_master_param and param_state["exp_avg"].
                        dtype != torch.float32:
                        param_state["exp_avg"] = param_state["exp_avg"
                            ].float()
                        param_state["exp_avg_sq"] = param_state["
                            exp_avg_sq"].float()

                    exp_avg, exp_avg_sq = param_state["exp_avg"],
                        param_state["exp_avg_sq"]
                    param_state["step"] += 1

                    exp_avg.mul_(betas[0]).add_(grad_data, alpha=1 -
                        betas[0])
                    exp_avg_sq.mul_(betas[1]).addcmul_(grad_data,
                        grad_data, value=1 - betas[1])

                    bias_correction1 = 1 - betas[0] ** param_state["
                        step"]
                    bias_correction2 = 1 - betas[1] ** param_state["
                        step"]
                    step_size = lr * math.sqrt(bias_correction2) /
                        bias_correction1

                    denom = exp_avg_sq.sqrt().add_(eps)
                    adam_update = exp_avg / denom

                    if weight_decay > 0:
                        param_data.mul_(1 - step_size * weight_decay)

                    param_data.add_(adam_update, alpha=-step_size)

                    if use_master_param:
                        p.data.copy_(param_data.to(dtype=p.data.dtype)
                            )

        return loss


def get_nora_optimizer(
    model,
    lr_nora=0.005,
    lr_adam=0.001,
    weight_decay=0.1,
```

```
    momentum=0.95,
    beta=0.95,
):
    nora_params = []
    adam_params = []

    for name, param in model.named_parameters():
        if param.requires_grad:
            if param.ndim >= 2 and "embed" not in name and "lm_head"
                not in name:
                nora_params.append(param)
            else:
                adam_params.append(param)

    param_groups = [
        dict(
            params=nora_params,
            lr=lr_nora,
            lr_nora=lr_nora,
            lr_adam=lr_adam,
            weight_decay=weight_decay,
            momentum=momentum,
            beta=beta,
            is_nora=True,
        ),
        dict(
            params=adam_params,
            lr=lr_adam,
            lr_nora=lr_nora,
            lr_adam=lr_adam,
            weight_decay=weight_decay,
            momentum=momentum,
            beta=beta,
            is_nora=False,
        ),
    ]
    optimizer = Nora(param_groups)
    return optimizer
```