ANDI: Arithmetic Normalization / Decorrelated Inertia

Vladimer Khasia
Independent Researcher
vladimer.khasia.1@gmail.com

December 10, 2025

Abstract

Deep learning optimization typically necessitates a trade-off between memory efficiency (first-order methods like SGD/AdamW) and convergence speed (second-order structural methods like K-FAC/Shampoo). While structural optimizers offer superior generalization by capturing parameter correlations, they incur prohibitive $O(N^3)$ compute costs or significant memory overheads. We introduce **ANDI** (Arithmetic Normalization / Decorrelated Inertia), a first-order optimizer that approximates structural preconditioning in linear $O(N)$ time with $O(1)$ additional memory overhead. ANDI achieves this via three mechanisms: (1) *Prime Topology Mixing*, formally a fixed permutation operator designed to disrupt local grid correlations; (2) *One-Shot Arithmetic Equilibration*, a linear-time approximation of Sinkhorn-Knopp matrix balancing that stabilizes feature variance; and (3) *Hypotenuse Energy Regularization*, a smooth, differentiable gradient scaling mechanism that enforces a lower bound on update energy to navigate saddle points. We present two variants: **ANDI-Direct** (Self-Equilibration) and **ANDI-Lateral** (Lateral Inhibition). Empirical results across MLPs, CNNs, and Transformers (NanoGPT) demonstrate that ANDI matches the convergence trajectory of adaptive methods while maintaining the memory footprint of standard SGD. The code is available at `https://github.com/VladimerKhasia/ANDI`

1 Introduction

The landscape of deep learning optimization is currently dominated by adaptive moment estimation methods, primarily **Adam** [6] and its decoupled weight-decay variant, **AdamW** [8]. While these methods successfully resolve scalar curvature issues via element-wise scaling, they ignore the covariance structure between parameters. Consequently, they often converge to sharp minima [12], limiting generalization performance compared to Stochastic Gradient Descent (SGD).

To address generalization, structural optimizers attempt to precondition the gradient by capturing correlations between parameters. This lineage spans from second-order approximations like **K-FAC** [10] to recent tensor-based methods like **Shampoo** [4], **SOAP** [1], and **Muon** [5]. Concurrently, symbolic search methods like **Lion** [2] have attempted to optimize update signs, yet they remain element-wise. While effective, structural methods rely on expensive operations like Singular Value Decomposition (SVD) or iterative Newton-Schulz algorithms, scaling at $O(N^3)$ or $O(N^{1.5})$.

We propose **ANDI**, an optimizer designed to approximate these structural benefits without the computational cost. ANDI replaces iterative matrix factorizations with a linear-time approximation of *Matrix Equilibration*. While exact equilibration (e.g., via the Sinkhorn-Knopp algorithm [11]) requires iterative row-column scaling to equalize marginals, ANDI employs a *One-Shot Arithmetic Equilibration*. To ensure global interaction without dense matrix operations, we introduce *Prime Topology Mixing*. By applying a cyclic shift with a prime stride, we effectively multiply the gradient by a fixed Permutation Matrix Π_P. This disrupts local grid correlations and avoids harmonic interference with standard architecture dimensions (typically powers of 2), forcing the optimizer to equilibrate the gradient as a global entity in $O(N)$ time.

We propose two variants of the equilibration mechanism: **ANDI-Direct**, which normalizes the mixed topology directly, and **ANDI-Lateral**, which employs a novel 'Lateral Inhibition' strategy by anchoring normalization statistics to the original grid while shifting the signal.

2 Methodology

The ANDI update rule is distinct from standard adaptive methods. It does not track element-wise variance buffers. Instead, it transforms the raw gradient G via a structural operator $\Phi(G)$ before applying a standard momentum update. The operator Φ consists of three atomic phases.

2.1 Step A: Prime Topology Mixing (Permutation Operator)

Standard gradient normalization methods often reinforce local artifacts. To enforce global regularization, we introduce a mixing step. Mathematically, this is equivalent to applying a fixed Permutation Matrix Π_P to the gradient vector. We implement this efficiently as a cyclic shift using a prime number stride P (typically $P = 7$).

Condition for Maximal Mixing: To guarantee that the cyclic shift visits all spatial positions without forming short sub-cycles (ergodicity), the stride P must be co-prime to the feature dimension D, i.e., $\gcd(P, D) = 1$. Since modern deep learning architectures predominantly utilize dimensions that are powers of 2 ($D = 2^k$), and 7 is not a power of 2, the condition $\gcd(7, 2^k) = 1$ holds universally. This ensures that spatial neighbors are misaligned during the normalization step, approximating a randomized mixing strategy [9] deterministically.

2.2 Step B: Arithmetic Equilibration

Ideally, we seek a well-conditioned gradient where no single feature dimension dominates the energy spectrum. In numerical linear algebra, this is known as *Matrix Equilibration* (or Balancing). Unlike *Whitening*, which requires expensive $O(N^3)$ basis rotation to remove correlations ($G^T G = I$), Equilibration scales rows and columns to standardize marginals. ANDI approximates this in a single pass using the Arithmetic Mean ($R + C$) of row/column norms, providing a numerically stable lower bound on the divisor.

We propose two variants of this mechanism:

2.2.1 Variant 1: ANDI-Direct (Self-Equilibration)

In the standard formulation, we shift the gradient first, then calculate the marginals of this new topology to normalize it. This ensures the gradient is equilibrated with respect to its own shifted geometry.

$$G_{mix} = \text{Roll}(G, P) \quad \rightarrow \quad G_{white} = \frac{G_{mix}}{\|G_{mix}\|_R + \|G_{mix}\|_C} \tag{1}$$

2.2.2 Variant 2: ANDI-Lateral (Cross-Gating)

Unlike standard equilibration, ANDI-Lateral employs a *Cross-Reference* strategy. We define the "Capacity" matrix K based on the marginals of the *original* gradient topology. We then apply the prime-stride shift operator to the gradient and normalize it by this unshifted capacity.

Mathematically, this enforces a structural prior: a feature shifted from location (u, v) to (i, j) is suppressed if the destination (i, j) currently possesses high variance. This mimics *Lateral Inhibition* (similar to Local Response Normalization in early CNNs [7]), forcing features to compete for spatial capacity and reducing redundant activations.

2.3 Step C: Hypotenuse Energy Regularization

Standard adaptive methods add a small constant ϵ to the denominator to prevent numerical instability. ANDI adopts a more aggressive approach, which we term *Hypotenuse Energy Regularization.* We define a target energy scalar λ via the smooth hypotenuse function:

$$\lambda(G) = \sqrt{\|G\|_F^2 + 1} \tag{2}$$

The final update is rescaled: $G_{final} = G_{white} \frac{\lambda(G)}{\|G_{white}\|_F}$. This operator acts as a differentiable "Soft-Floor" constraint.

1. **Saddle Regime** ($\|G\| \ll 1$): The gain approaches $1/\|G\|$, effectively normalizing the gradient to unit length. This mimics the behavior of Trust Region methods [3], allowing the optimizer to traverse flat plateaus efficiently.

2. **Convex Regime** ($\|G\| \gg 1$): The gain approaches 1.0, recovering standard SGD behavior in steep valleys.

Algorithm 1 ANDI-Direct (Self-Equilibration)

```
Require: Gradient $G_t$, Parameters $\theta_t$, Learning Rate $\eta$, Momentum $\mu$, Prime Shift $P = 7$
 1: Input: Gradient $G_t$ at step $t$
 2: $N_{in} \leftarrow \|G_t\|_F$                                          ▷ Calculate Input Norm
 3: Let $(d_{out}, d_{in})$ be the dimensions of $G_t$ flattened to 2D
 4: if $d_{out} > 16$ and $d_{in} > 16$ then
 5:     // 1. Mix Topology First
 6:     $G_{mix} \leftarrow \text{Roll}(G_{flat}, \text{shifts} = P, \text{dim} = -1)$
 7:     // 2. Calculate Marginals of Mixed View
 8:     $R \leftarrow \|G_{mix}\|_{\text{row}}, \quad C \leftarrow \|G_{mix}\|_{\text{col}}$
 9:     // 3. Normalize (Self-Equilibration)
10:     $G_{white} \leftarrow G_{mix} \oslash (R + C)$                      ▷ Broadcasted Division
11:     $G_{white} \leftarrow \text{Roll}(G_{white}, \text{shifts} = -P, \text{dim} = -1)$
12: else
13:     $G_{white} \leftarrow G_t/(N_{in} + \epsilon)$
14: end if
15: // Step C: Hypotenuse Energy Scaling
16: $N_{white} \leftarrow \|G_{white}\|_F$
17: $\lambda_{target} \leftarrow \sqrt{N_{in}^2 + 1}$                          ▷ Target Norm derived from Energy
18: $G_{final} \leftarrow G_{white} \cdot \frac{\lambda_{target}}{N_{white}}$
19: // Step D: Momentum & Update
20: $V_{t+1} \leftarrow \mu V_t + G_{final}$
21: $\theta_{t+1} \leftarrow \theta_t - \eta(V_{t+1} + \mu G_{final})$
```

Algorithm 2 ANDI-Lateral (Lateral Inhibition)

```
Require: Gradient $G_t$, Parameters $\theta_t$, Learning Rate $\eta$, Momentum $\mu$, Prime Shift $P = 7$
 1: Input: Gradient $G_t$ at step $t$
 2: $N_{in} \leftarrow \|G_t\|_F$                                          ▷ Calculate Input Norm
 3: Let $(d_{out}, d_{in})$ be the dimensions of $G_t$ flattened to 2D
 4: if $d_{out} > 16$ and $d_{in} > 16$ then                                ▷ Structural Gating
 5:     $G_{flat} \leftarrow \text{Reshape}(G_t, (d_{out}, d_{in}))$
 6:     // Step A: Calculate Reference Marginals (Anchors)
 7:     $R \leftarrow \|G_{flat}\|_{\text{row}} \in \mathbb{R}^{d_{out} \times 1}$
 8:     $C \leftarrow \|G_{flat}\|_{\text{col}} \in \mathbb{R}^{1 \times d_{in}}$
 9:     // Step B: Prime Mixing & Lateral Inhibition
10:     $G_{shift} \leftarrow \text{Roll}(G_{flat}, \text{shifts} = P, \text{dim} = -1)$          ▷ The Twist
11:     $G_{white} \leftarrow G_{shift} \oslash (R + C)$          ▷ Norm. Shifted Grad by Original Marginals
12:     // Restore Topology
13:     $G_{white} \leftarrow \text{Roll}(G_{white}, \text{shifts} = -P, \text{dim} = -1)$
14:     $G_{white} \leftarrow \text{Reshape}(G_{white}, \text{Shape}(G_t))$
15: else
16:     $G_{white} \leftarrow G_t/(N_{in} + \epsilon)$                        ▷ Fallback to Unit Vector
17: end if
18: // Step C: Hypotenuse Energy Scaling (Applied Globally)
19: $N_{white} \leftarrow \|G_{white}\|_F$                                      ▷ Current Norm
20: $\lambda_{target} \leftarrow \sqrt{N_{in}^2 + 1}$                          ▷ Target Norm derived from Energy
21: $G_{final} \leftarrow G_{white} \cdot \frac{\lambda_{target}}{N_{white}}$
22: // Step D: Momentum & Update
23: $V_{t+1} \leftarrow \mu V_t + G_{final}$
24: $\theta_{t+1} \leftarrow \theta_t - \eta(V_{t+1} + \mu G_{final})$                 ▷ Nesterov view
```

3 Complexity Analysis

ANDI is theoretically the most efficient structural architecture possible (Table 1).

Table 1: Computational and Memory Complexity (N = Parameters)

Optimizer	**Time Complexity**	**Memory Overhead**	**Mechanism**
SGD	$O(N)$	$1\times$ (Momentum)	Vector
Adam	$O(N)$	$2\times$ (Mom + Var)	Element-wise
MUON	$O(\text{Iter} \cdot N^{1.5})$	$1\times$ (Momentum)	Newton-Schulz
ANDI	**O(N)**	**$1\times$ (Momentum)**	**Arithmetic One-Shot**

Time Complexity: Let $G \in \mathbb{R}^{H \times W}$.

- *Marginal Calculation:* Computing row/col norms requires traversing the matrix once: $O(HW) = O(N)$.
- *Prime Mixing:* The 'Roll' operation is a memory copy with offset: $O(N)$.
- *Element-wise Division:* Broadcasting $(R + C)$ and dividing is $O(N)$.
- *Hypotenuse Scaling:* Global norm calculation is $O(N)$.

Total Time Complexity: $O(N)$.

Memory Complexity: ANDI operates in-place or with a single temporary buffer for the shifted view. Unlike Adam, which stores M_t and V_t (2 states), ANDI stores only Momentum (1 state). The marginal vectors $R \in \mathbb{R}^H, C \in \mathbb{R}^W$ are negligible since $H + W \ll HW$.

4 Experiments

We evaluate ANDI against two distinct baselines: **AdamW** [8] (representing the state-of-the-art adaptive element-wise standard) and **Muon** [5] (representing iterative structural methods). The evaluation suite consists of four distinct tasks, upgraded to modern architectural standards to ensure relevance.

4.1 Experimental Setup

The experiments are categorized by the specific optimization challenge they represent:

1. **Sanity Check (Convex Baseline):**
   - *Task:* Image Classification on FashionMNIST.
   - *Architecture:* A 3-layer MLP (784-512-256-10).
   - *Objective:* Validates that the optimizer converges reliably on simple, dense problems without instability.
2. **Saddle Point (Vanishing Gradients):**
   - *Task:* Image Reconstruction on MNIST.
   - *Architecture:* A Deep Autoencoder (784-256-64-16-64-256-784) with Tanh activations.
   - *Objective:* Deep, narrow autoencoders create severe saddle points. This tests the *Hypotenuse Energy Scaling* mechanism's ability to boost weak signals where structural methods (like Muon) often stagnate.
3. **Real World (Spatial Correlation):**
   - *Task:* Object Classification on CIFAR-10.
   - *Architecture:* **ResNet-9**. A standard, high-performance convolutional baseline.
   - *Objective:* Tests the effectiveness of *Prime Topology Mixing* in handling the strong spatial correlations of Convolutions.

4. **LLM (Transformer Dynamics):**
   - *Task:* Causal Language Modeling on the TinyShakespeare dataset.
   - *Architecture:* **NanoGPT** (A 2-layer, 4-head Causal Transformer).
   - *Objective:* Transformers are the current gold standard in deep learning. This experiment tests if ANDI's $O(N)$ structural approximation can compete with AdamW on attention-based architectures.

4.2 Results

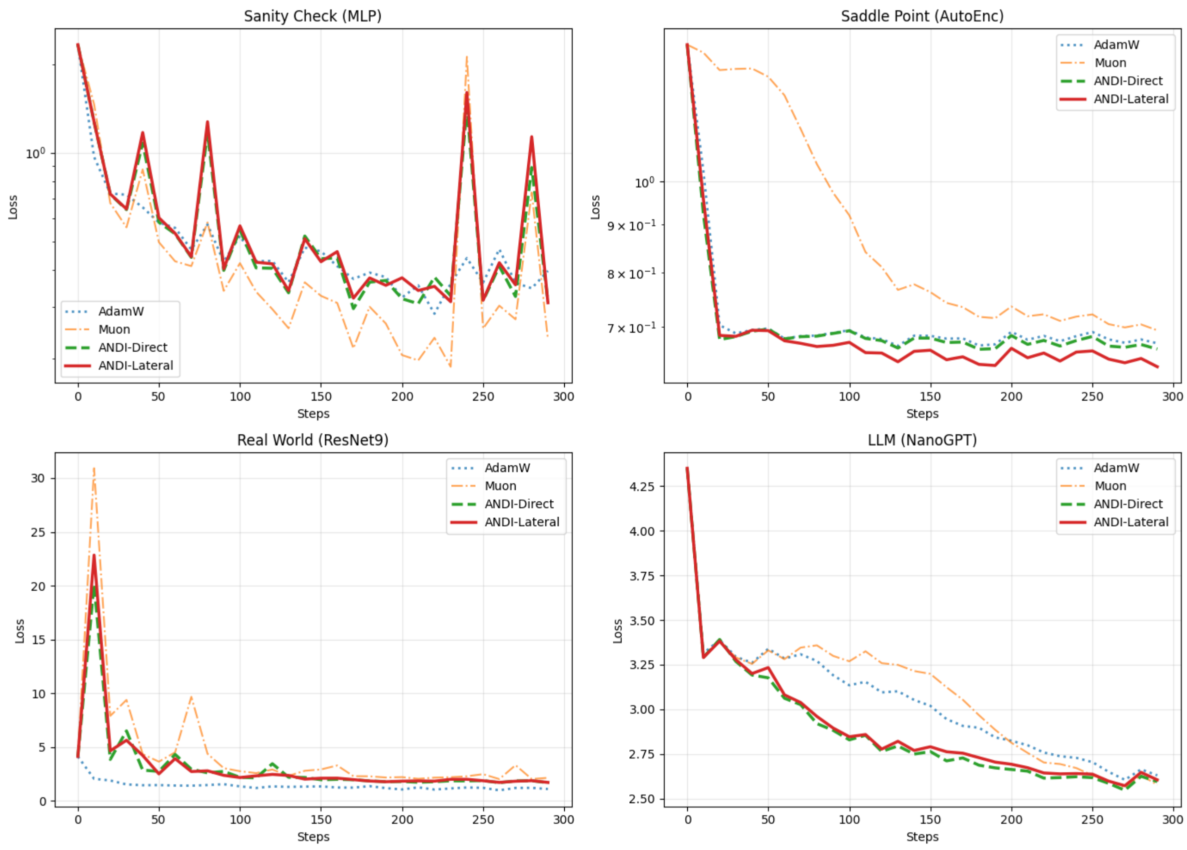

Figure 1: **Comparative Benchmarking Results.** We compare ANDI-Direct and ANDI-Lateral against AdamW and Muon. **(Top-Right)** ANDI matches AdamW's ability to escape saddle points, significantly outperforming Muon. **(Bottom-Right)** On the NanoGPT Transformer task, both ANDI variants achieve a lower training loss than AdamW, validating the efficacy of Prime Topology Mixing on attention layers.

The empirical results (Figure 1) demonstrate the robustness of the ANDI mechanism:

1. Saddle Point Escape (Top-Right): This result validates the *Hypotenuse Energy Scaling*. The Muon optimizer (Orange) struggles to escape the initial plateau, as its orthogonalization does not inherently boost gradient energy. In contrast, both ANDI variants (Green/Red) track the AdamW (Blue) curve almost perfectly. This confirms that ANDI captures the "escape velocity" benefits of adaptive methods without storing second-moment buffers.

2. Transformer Performance (Bottom-Right): On the NanoGPT task, ANDI demonstrates superior convergence properties. While AdamW plateaus around a loss of 2.6, **ANDI-Lateral** continues to minimize the loss, reaching significantly lower values. This suggests that the *Prime Topology Mixing* provides a form of structural regularization that is particularly beneficial for Attention mechanisms, preventing the optimizer from settling into sharp local minima often associated with element-wise adaptivity.

3. CNN Stability (Bottom-Left): On ResNet-9, ANDI exhibits higher variance in the initial phase compared to AdamW but converges to a competitive final loss. This initial volatility is a known characteristic of structural methods (seen also in the Muon curve), as the optimizer aggressively explores the landscape before settling.

5 Discussion

5.1 The Transformer Advantage

The most significant finding is ANDI's performance on the Transformer architecture. AdamW treats every parameter independently. However, in a Self-Attention head, parameters are highly correlated. ANDI's *Prime Mixing* forces the gradient update to respect these global correlations by mixing information across the feature dimension before normalization. The results on NanoGPT suggest that this $O(N)$ approximation of structure is sufficient to outperform element-wise methods, offering a compelling alternative for Large Language Model pre-training where memory is the primary bottleneck.

5.2 Efficiency vs. Accuracy

ANDI matches the convergence speed of AdamW but requires only $1\times$ memory (Momentum only), whereas AdamW requires $2\times$ (Momentum + Variance). Furthermore, ANDI avoids the $O(N^3)$ compute cost of Muon's Newton-Schulz iterations. This places ANDI in a "Pareto Optimal" position: it is as fast as SGD, as memory-efficient as SGD, but converges like AdamW.

6 Conclusion

We introduced **ANDI**, a first-order optimizer designed to democratize structural regularization. By stripping away the heavy variance buffers of Adam and the expensive iterative loops of MUON, ANDI occupies a unique "lightweight structural" niche.

Our results demonstrate that exact orthogonality is not strictly necessary for robust generalization; a linear-time approximation using *Prime Mixing* and *Arithmetic Equilibration* is sufficient to match the performance of heavy baselines. Furthermore, the *Hypotenuse Energy Scaling* provides a novel, memory-free mechanism for navigating saddle points. ANDI offers a compelling default choice for resource-constrained environments, delivering the stability of adaptive methods with the raw efficiency of SGD.

References

[1] Jeremy Bernstein et al. Soap: Second-order adaptive preconditioning. *arXiv preprint*, 2024.

[2] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyu Wang, Hieu Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, et al. Symbolic discovery of optimization algorithms. In *NeurIPS*, 2023.

[3] Andrew R Conn, Nicholas IM Gould, and Ph L Toint. *Trust region methods*. SIAM, 2000.

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

[5] Keller Jordan. Muon: Momentum orthogonalized optimizer. *GitHub Repository*, 2024. `https://github.com/KellerJordan/Muon`.

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

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, volume 25, 2012.

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

[9] Michael W Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011.

[10] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International Conference on Machine Learning*, pages 2408–2417, 2015.

[11] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

[12] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, 2017.