SaMA: Morpho Adaptation via Asymmetric Expansion of Kronecker Product

An Nguyen
Department of Artificial Intelligence
Korea University
nguyenan@korea.ac.kr

Anh Tong
Department of Artificial Intelligence
Korea University
anhtong12@korea.ac.kr

Abstract

Low-rank adaptation (LoRA) has attracted significant attention in the parameter-efficient fine-tuning (PEFT) landscape. However, its design suffers from two fundamental limitations. First, its invariance manifold induces flat directions in the loss landscape, creating an optimization bottleneck for adaptive optimizers. Second, its effective rank scales only *linearly* with the parameter budget. In this paper, we propose *Scalable Morpho Adaptation (SaMA)*, a PEFT method grounded in a principled generalization of the Kronecker product. By relaxing the shared-block constraint in the perfect-shuffle factorization, SaMA spans from low to full rank and achieves a practical rank that scales *quadratically* with the parameter budget, up to full-rank. Furthermore, we show that its invariance manifold collapses to a diagonal group, making it structurally more optimizer-friendly. Empirically, SaMA consistently outperforms strong PEFT baselines on commonsense and arithmetic reasoning benchmarks while being more parameter-efficient.

1 Introduction

Parameter-efficient fine-tuning (PEFT) has become the standard approach for adapting large pre-trained foundation models [1, 2, 3, 4, 5, 6] to downstream tasks. Among existing methods, low-rank adaptation (LoRA) [7] is the most widely adopted, and a growing body of research work seeks to improve and extend it [8, 9, 10, 11, 12, 13]. Yet, despite its practical success, LoRA suffers from two structural limitations that grow more severe as the rank increases.

Optimization pathology from gauge freedom. Any factorization $\Delta\mathbf{W} = \mathbf{BA}$ is invariant under the reparameterization $(\mathbf{BC})(\mathbf{C}^{-1}\mathbf{A})$ for any invertible $\mathbf{C}$, defining an r^2-dimensional manifold of equivalent solutions, which is not restricted to rescaling individual latent channels, but contains rotations and shears mixing them. This invariance introduces flat directions in the loss landscape that are poorly handled by adaptive optimizers such as Adam [14, 15], whose per-coordinate preconditioner implicitly assumes axis-aligned curvature. The resulting symptoms are rank collapse and growing hyperparameter sensitivity at higher ranks [16].

Linear cost for limited expressivity. A rank-r LoRA adapter spans only r independent directions of weight change. When a task requires richer updates, increasing r scales the parameter count *linearly* without a commensurate gain in structural expressivity. Recent high-rank variants [17, 11, 18] attempt to bridge this gap, but typically incur their own costs including materializing large intermediate matrices [19] or relying on deep tensor chains [11].

In this paper, we revisit sparse structured matrices for PEFT through a generalization of the Kronecker product. The key observation is that $\mathbf{A} \otimes \mathbf{B}$ can be rewritten as the product of two block-diagonal matrices interleaved by a perfect-shuffle permutation. Relaxing the shared-block constraint of this factorization yields a strictly richer family of updates, which we call the *Morpho* family, and which forms the foundation of *Scalable Morpho Adaptation (SaMA)*. The adapter rank in SaMA is governed

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

by the block sizes, allowing it to reach full rank while remaining sparse throughout. Crucially, we prove that the gauge group of SaMA collapses from $GL(r)$-group invariance to a group of *diagonal scalings* that acts on each channel separately. This thereby eliminates the pathological flat directions that hinder Adam-style optimizers. Empirically, we evaluate SaMA on natural language reasoning benchmarks—including reading comprehension and arithmetic reasoning. Our method consistently outperforms strong baselines across model families and parameter budgets.

Our contributions are threefold: **(1)** We propose SaMA, a PEFT method derived from a structural generalization of the Kronecker product via Morpho block-diagonal factorization, supporting a broad spectrum of ranks from low to full rank with high parameter efficiency. **(2)** We analyze the rank, scaling, and gauge properties of SaMA, showing *quadratic* rank growth with respect to the parameter count, stable scaling behavior, and a diagonal invariant manifold that is inherently more optimizer-friendly than that of LoRA, contains no channel-mixing transformations. **(3)** We demonstrate strong empirical results across diverse reasoning tasks, showing that the structural expressivity of SaMA offers a robust and favorable alternative to existing PEFT methods.

2 Methodology

This section reviews the Kronecker product and Monarch parameterization, establishes a structural connection between them, and shows how that connection motivates the design of Morpho adapters. Building on this formulation, we provide theoretical analyses of rank flexibility, scaling stability, and parameter efficiency, followed by a discussion of practical implementation.

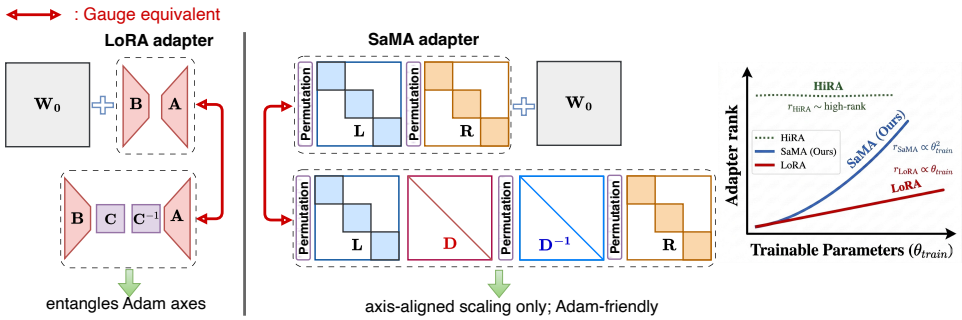

Figure 1: Gauge symmetry in LoRA vs. Morpho adapter parameterizations. *(Left)* LoRA's update $\Delta\mathbf{W} = \mathbf{BA}$ admits any invertible $\mathbf{C} \in GL(r)$ as a gauge transform, including rotations and shearing. *(Center)* The block-diagonal factorization $\boldsymbol{P}_l^\top \mathbf{L} \boldsymbol{P}_r \mathbf{R}$ reduces the gauge freedom to diagonal rescalings only: $\mathbf{L} \leftarrow \mathbf{LD}$ and $\mathbf{R} \leftarrow \boldsymbol{P}_r^\top \mathbf{D}^{-1} \boldsymbol{P}_r \mathbf{R}$ for diagonal $\mathbf{D}$. Unlike LoRA, this coordinate-aligned scaling does not entangle latent features. *(Right)* Expressivity: the rank of a SaMA adapter scales as a quadratic function of the training budget.

2.1 Kronecker Product

The Kronecker product [20], denoted $\otimes$, takes two matrices $\mathbf{A} \in \mathbb{R}^{m_1 \times n_1}$ and $\mathbf{B} \in \mathbb{R}^{m_2 \times n_2}$ and produces $\mathbf{C} = \mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{M \times N}$, where $M = m_1 m_2$ and $N = n_1 n_2$:

$$\mathbf{C} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n_1}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m_1 1}\mathbf{B} & \cdots & a_{m_1 n_1}\mathbf{B} \end{bmatrix}$$

For an input vector $\mathbf{x} \in \mathbb{R}^N$, define the matrix views $\mathbf{X} = \mathrm{unvec}(\mathbf{x})$ and $\mathbf{Y} = \mathrm{unvec}(\mathbf{y})$, where $\mathrm{vec}(\cdot)$ stacks columns into a vector and $\mathrm{unvec}(\cdot)$ reverses this operation. Applying the Kronecker product as a linear map admits the equivalent forms:

$$\mathbf{y} = (\mathbf{A} \otimes \mathbf{B})\,\mathbf{x} \;\Leftrightarrow\; \mathbf{Y} = \mathbf{B}\,\mathbf{X}\,\mathbf{A}^\top \;\Leftrightarrow\; \mathbf{y} = \mathrm{vec}\big(\mathbf{B}\ \mathrm{unvec}(\mathbf{x})\,\mathbf{A}^\top\big). \tag{1}$$

The matrix form in Eq. (1) avoids materializing the full $M \times N$ product and is efficient in practice.

2.2 Monarch Parameterization for Square Matrices

Definition 2.1. *[21] Let $N = n^2$. A* square Monarch matrix $\mathbf{M} \in \mathbb{R}^{N \times N}$ *has the form:*

$$\mathbf{M} = \boldsymbol{P}^\top \mathbf{L}\, \boldsymbol{P} \mathbf{R},$$

where $\mathbf{L} = \mathrm{diag}(\mathbf{L}_1, \ldots, \mathbf{L}_n)$ *and* $\mathbf{R} = \mathrm{diag}(\mathbf{R}_1, \ldots, \mathbf{R}_n)$ *are block-diagonal matrices each comprising* n *dense blocks of size* $n \times n$*, and* $\boldsymbol{P}$ *is the permutation matrix that transposes the two indices of an* $n \times n$ *matrix obtained by reshaping a vector of length* N*.*

Monarch parameterization [21] is an extension of butterfly matrices [22, 23], representing a matrix as the product of two block-diagonal matrices interleaved by permutations.

2.3 Morpho: Bridging the Kronecker Product and Monarch Parameterization

We now introduce the *Morpho matrix*, a generalization of both the Kronecker product and Monarch matrices, designed for scalable, omni-rank PEFT.

2.3.1 Kronecker Product as a Two-Stage Sparse Factorization

We analyze the structure of $\mathbf{A} \otimes \mathbf{B}$ by decomposing it into two sequential sparse transformations. Applying the mixed-product property of the Kronecker product [24, 25], $(\mathbf{A} \otimes \mathbf{E})(\mathbf{F} \otimes \mathbf{B}) = (\mathbf{AF}) \otimes (\mathbf{EB})$, and setting $\mathbf{E} = \mathbf{I}_{m_2}$, $\mathbf{F} = \mathbf{I}_{n_1}$ yields:

$$\mathbf{A} \otimes \mathbf{B} = \underbrace{(\mathbf{A} \otimes \mathbf{I}_{m_2})}_{\mathsf{L}'} \underbrace{(\mathbf{I}_{n_1} \otimes \mathbf{B})}_{\mathsf{R}}. \tag{2}$$

This reveals a two-stage structure: the right factor R maps the input from dimension $N = n_1 n_2$ to a hidden dimension $n_1 m_2$, and the left factor L' maps the result to the output dimension $M = m_1 m_2$.

Concretely, $\mathsf{R} = \mathbf{I}_{n_1} \otimes \mathbf{B} = \mathrm{diag}(\mathbf{B}, \ldots, \mathbf{B})$ is block-diagonal with n_1 identical copies of $\mathbf{B}$. The left factor L', however, is *not* block-diagonal: its $m_1 \times n_1$ blocks of size $m_2 \times m_2$ are scaled identities (with scalar entries drawn from $\mathbf{A}$), as shown above.

$$\mathsf{L}' = \begin{bmatrix} \mathbf{D}_{11} & \cdots & \mathbf{D}_{1m_2} \\ \vdots & \ddots & \vdots \\ \mathbf{D}_{m_2 1} & \cdots & \mathbf{D}_{m_2 m_2} \end{bmatrix}$$

To expose the hidden block-diagonal structure of L', we apply the *perfect shuffle* permutation [24] (see Appendix A), which reshapes a vector of length r into a $p \times q$ matrix in row-major order, transposes it, and re-vectorizes. Using the identity $\mathbf{A} \otimes \mathbf{I}_{m_2} = \boldsymbol{P}_{m_1,m_2}^\top (\mathbf{I}_{m_2} \otimes \mathbf{A}) \boldsymbol{P}_{n_1,m_2}$, we rewrite:

$$\mathsf{L}' = \boldsymbol{P}_l^\top \mathbf{L} \boldsymbol{P}_r, \qquad \mathbf{L} = \mathrm{diag}(\mathbf{A}, \ldots, \mathbf{A}), \quad \boldsymbol{P}_l^\top = \boldsymbol{P}_{m_1,m_2}^\top, \quad \boldsymbol{P}_r = \boldsymbol{P}_{n_1,m_2}. \tag{3}$$

Equation (3) reveals that both factors of the Kronecker product share the same block-diagonal structure, up to a pair of perfect shuffle permutations. This symmetric view directly motivates the generalization introduced in the next section.

2.3.2 Generalizing the Kronecker Decomposition

In the Kronecker product, all diagonal blocks within each factor are *tied* — every block equals either $\mathbf{A}$ or $\mathbf{B}$. Releasing this constraint and allowing each block to be an *independent* dense matrix yields a strictly richer family of transformations while preserving the computational advantages of block-diagonal structure.

Fine-tuning via reparameterization. Given a frozen pre-trained weight $\mathbf{W}_0 \in \mathbb{R}^{M \times N}$, we introduce a trainable additive update $\Delta\mathbf{W}$ and define the fine-tuned forward pass as:

$$\mathbf{W}_{\mathrm{ft}} = \mathbf{W}_0 + \Delta\mathbf{W}, \qquad \mathbf{y} = \mathbf{W}_0 \mathbf{x} + \Delta\mathbf{W} \mathbf{x}.$$

The design of $\Delta\mathbf{W}$ is the central contribution of SaMA.

Scalable Morpho Adaptation (Morpho). Motivated by the Kronecker decomposition in Sec. 2.3.1 and the Monarch structure in Sec. 2.2, we parameterize the update as a *Morpho matrix*:

$$\Delta\mathbf{W} = s_{\mathrm{SaMA}} \boldsymbol{P}_l^\top \mathbf{L} \boldsymbol{P}_r \mathbf{R}, \tag{4}$$

where $\mathbf{L}$ and $\mathbf{R}$ are block-diagonal matrices with *independent* dense blocks,

$$\mathbf{L} = \mathrm{diag}(\mathbf{L}_1, \ldots, \mathbf{L}_{m_2}), \qquad \mathbf{R} = \mathrm{diag}(\mathbf{R}_1, \ldots, \mathbf{R}_{n_1}), \tag{5}$$

with $\mathbf{L}_i \in \mathbb{R}^{m_1 \times n_1}$ and $\mathbf{R}_j \in \mathbb{R}^{m_2 \times n_2}$ each independently parameterized, and $n_1, m_2 \ll \min(M, N)$. The permutations $\boldsymbol{P}_l = \boldsymbol{P}_{m_1,m_2}$ and $\boldsymbol{P}_r = \boldsymbol{P}_{n_1,m_2}$ are fixed perfect shuffles (see Eq. (10)) and contribute no trainable parameters. The scalar s_{SaMA} controls the update magnitude;

its role in training stability is analyzed in Sec. 3.3. We index a specific adapter configuration by the tuple (n_1, m_2), written $\text{SaMA}_{n_1 \times m_2}$.

We denote the set of all matrices expressible in the form of Eq. (4) as $\mathcal{F}^{(M,N)}$ (abbreviated $\mathcal{F}^{(N)}$ for square matrices), and the set of all compatible factor pairs $(\mathbf{L}, \mathbf{R})$ satisfying Eq. (5) as $\mathcal{F}_g$.

Remark. The permutation $\boldsymbol{P}_{n_1,n_2}$ maps a vector $[x_1, \ldots, x_N]$ to $[x_1, x_{1+n_2}, \ldots, x_{1+(n_1-1)n_2}, x_2, x_{2+n_2}, \ldots, x_{n_2}, x_{2n_2}, \ldots, x_{n_1 n_2}]$, equivalent to reading an $n_1 \times n_2$ matrix in column-major order after writing it in row-major order. When $M = N = n^2$ and all blocks are square, Eq. (4) reduces exactly to the Monarch parameterization of Sec. 2.2.

Discussion. SaMA differs fundamentally from Dense-to-Sparse fine-tuning (D2S-FT) [21], which applies Monarch *square matrices* in pre-training and requires projecting weights into sparse form before training. SaMA instead operates purely in the additive adapter regime, requiring no such pre-processing. Critically, it spans low-rank to full-rank updates within a single parameterization, addressing both the low intrinsic dimensionality of fine-tuning landscapes [26, 27] and the broader need for omni-rank adaptation, which establishes a new baseline alongside the LoRA family [7, 28, 8] and recent high-rank methods [11, 29].

Comparison with block-diagonal methods. SaMA is architecturally distinct from MoRE [30] and similar block-diagonal adapters. MoRE partitions the weight matrix spatially into b_M decoupled blocks, in both left and right matrices, like a collection of low-rank adapters, which results in a risk of sparse adapter. SaMA is instead constructed via the extended Kronecker product: the factor dimensions (n_1, m_2) are not spatial slices but intrinsically coupled tensor indices, influencing block counts. This mutually dependent structure guarantees dense output and only allows diagonal entanglement in parameter space degeneracy. (See Appendix D.7)

SaMA follows the reparameterization convention of LoRA [7] that $\mathbf{R}$ is initialized from a small-variance Gaussian and $\mathbf{L} = 0$, ensuring $\Delta\mathbf{W} = 0$ at the start of training with non-vanishing gradients. At inference time, the adapter merges into $\mathbf{W}_0$ with zero overhead, and forward passes through $\Delta\mathbf{W}$ during training cost $O(\nu N)$ FLOPs via two batched block-matrix multiplies — with no Dense-to-Sparse pre-processing required. Full implementation details are given in Appendix C.

3 Theoretical Analysis of SaMA

This section aims to examine several aspects of SaMA, including rank versatility in different regimes, adapter expressivity, numerical stability and gauge symmetry in parameter space; before establishing its theoretical properties.

3.1 Rank Properties

First, we analyze SaMA adapters and establish the theoretical property regarding the rank flexibility under random initialization.

Theorem 3.1 (Rank upper bound)**.** *Let* $\mathbf{L}$ *and* $\mathbf{R}$ *be block-diagonal matrices as defined in Eq.* (5)*, with* $n_1 m_2 \leq \min(M, N)$*, and let* $s \neq 0$*. Suppose every entry of each submatrix is drawn i.i.d. from a continuous distribution. Then, with probability* 1*:*

$$\text{rank}(\Delta\mathbf{W}) = \text{rank}\big(s\, \boldsymbol{P}_l^\top \mathbf{L} \boldsymbol{P}_r \mathbf{R}\big) = n_1 m_2. \tag{6}$$

Theorem 3.1 shows that the effective rank of the adapter is exactly $n_1 m_2$, determined jointly by the block counts of $\mathbf{L}$ and $\mathbf{R}$. This decoupling is structurally significant: by varying n_1 and m_2 independently, SaMA covers a broad range of rank regimes, from low-rank to full-rank adaptation. Notably, for a square $N \times N$ weight with N a perfect square, a full-rank update is achievable with only $2N\sqrt{N}$ parameters ($n_1 = m_2 = \sqrt{N}$).

To make the comparison concrete, consider a fixed parameter budget of $P = 2Nr$ shared by both methods. LoRA allocates this as two matrices of sizes $N \times r$ and $r \times M$, yielding adapter rank $\mathcal{R}_{\text{LoRA}} = r$, which grows *linearly* with P. SaMA with configuration $(n_1, m_2) = (r, r)$ matches the same budget while achieving adapter rank $\mathcal{R}_{\text{SaMA}} = n_1 m_2 = r^2$, growing *quadratically*.

3.2 Expressivity of Morpho

The Eckart-Young theorem [31] establishes rank-r matrices as the *optimal* Frobenius approximators of any target, given a budget of r singular components. This optimality is the cornerstone for

understanding the expressivity of LoRA and its variants. Here, we study the following question: *"Under what conditions can a higher-rank structure, specifically Morpho, surpass the Eckart-Young bound at the same parameter budget?"*

Theorem 3.2 (Expressivity crossover)**.** *Let $\mathbf{W} \in \mathbb{R}^{d\times d}$ be a rank-R matrix with exactly R unit singular values, and fix a parameter budget corresponding to $n_1 = m_2 = r < R$ in Eq. (4). Denote $\mathbf{W}^*_{LoRA}$ as the optimal rank-r approximation of $\mathbf{W}$ (the Eckart–Young truncation), and $\mathbf{W}^*_{Morpho}$ as the Morpho approximation of $\mathbf{W}$, under the same budget. Then there exists a threshold $R_0 < d$ such that, with probability 1:*

$$\|\mathbf{W} - \mathbf{W}^*_{Morpho}\|_F \;<\; \|\mathbf{W} - \mathbf{W}^*_{LoRA}\|_F, \qquad \forall\, R > R_0. \tag{7}$$

Theorem 3.2 identifies *effective rank* (eRank) [32] as the key discriminating quantity. At low eRank, $\mathbf{W}$'s spectrum is concentrated and Eckart–Young captures it nearly perfectly, whereas Morpho offers no benefit. At high eRank, the spectral mass of $\mathbf{W}$ is broadly distributed and the rank-r ceiling becomes binding. The Morpho structure covers a higher-dimensional subspace than the LoRA bottleneck because of its global mixing between diagonal blocks. The proof of Theorem 3.2 can be found in Appendix D.3.

Empirical validation. To justify Theorem 3.2, we design a controlled matrix-approximation stress test. For each parameterization class $\mathcal{S}_i$, we solve $\min_{\mathbf{W}'\in\mathcal{S}_i} \|\mathbf{W} - \mathbf{W}'\|_F$ on synthetic targets with systematically varied eRank. This setup deliberately isolates the approximation geometry from task signals, cleanly measuring how each parameterization covers the target manifold as the spectrum shifts. We deliberately design target matrices of size 1024×1024 with low entropy effective rank r, e.g. $\mathrm{erank}(\mathbf{W}) = \mathrm{rank}(\mathbf{W})$, to avoid spectrum decay in the low-rank target, then evaluate the performance on matrix approximation of LoRA and SaMA across a wide range of effective ranks. (See details in Appendix F.1.)

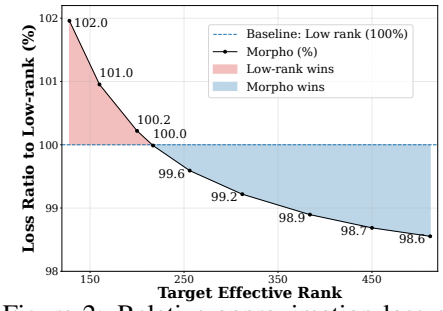

Figure 2: Relative approximation loss of Morpho vs. low-rank under matched parameter budgets. Morpho gains a decisive advantage beyond eRank $= 217$.

The results in Fig. 2 confirm the theoretical prediction. At low eRank, low-rank approximation is competitive or superior, consistent with Eckart–Young. Beyond eRank $= 217$, Morpho achieves strictly lower approximation loss under the same budget, with the gap widening monotonically. This result is precisely anticipated by Theorem 3.2.

3.3 Scaling stability

Theorem 3.3 (Numerical stability)**.** *The scaling factor s_{SaMA} that keeps the initialization variance of $\Delta\mathbf{W}$ stable after finite n SGD updates, as the hidden dimension n_1 grows, satisfies:*

$$s_{\mathrm{SaMA}} = \alpha\, n_1^{1/4} \quad\Longleftrightarrow\quad \Delta\mathbf{W} = \left(\alpha\, n_1^{1/4}\right) \boldsymbol{P}_l^\top \mathbf{L}\boldsymbol{P}_r\mathbf{R}, \qquad \alpha \in \tilde{\mathbb{R}}. \tag{8}$$

Equation (8) derives the scaling factor (unique up to a constant factor) that simultaneously stabilizes forward output variance and backward gradient magnitude of the trained adapter as n_1 grows. (See Appendix D.2 for the proof and the detailed explanation.) Using all other factors can lead the signal to either attenuate or diverge (empirically verified in Appendix D.4). Critically, this result is theoretically *derived*, unlike rsLoRA [33], which proposes $\alpha/\sqrt{r}$ empirically. To our knowledge, no existing block-structured adapter, including MoRE [30] and QuanTA [11] provides an analogous stability analysis.

3.4 Optimization dynamics in structured invariant manifolds

We also explore optimization behavior of structured adapter parameterizations. Particularly, we investigate how the geometry of invariant manifolds affects parameter space degeneracy and builds a connection with optimization.

Gauge symmetry in matrix factorization Any factorization $\Delta\mathbf{W} = \mathbf{BA}$ admits an infinite family of *equivalent* solutions. Inserting an invertible matrix $\mathbf{C} \in GL(r)$ yields $\Delta\mathbf{W} = (\mathbf{BC})(\mathbf{C}^{-1}\mathbf{A})$

with identical loss. This defines an *invariant manifold* indicating a continuous symmetry group under which the optimization objective is flat. For LoRA-style adapters, this manifold spans the full general linear group $GL(r)$, including rotations and shearing transformations in latent space.

The optimization bottleneck in invariance manifolds. Adaptive optimizers such as Adam [14, 15] operate coordinate-wise, implicitly assuming axis-aligned curvature. Rotational freedom in $GL(r)$ violates this assumption. That is, any rotation of the latent basis linearly recombines parameter columns of $\mathbf{B}$ and $\mathbf{A}$ and entangles features that Adam treats as independent [34]. Therefore, Adam cannot efficiently navigate the flat directions carved out by feature-mixing transformations.

We will show that Morpho parameterization does not suffer from this degeneracy, as shown by analyzing the invariant manifold of SaMA.

Theorem 3.4 (Invariance manifold of Morpho parameterization)**.** *Given a generic matrix* $\mathbf{M}_1 \in \mathcal{F}$: $\mathbf{M}_1 = \boldsymbol{P}_l^\top \mathbf{L}_1 \boldsymbol{P}_r \mathbf{R}_1$, *where* $\mathbf{L}_1, \mathbf{R}_1$ *are determined block-diagonal matrices. Let* $\mathcal{F}_g(\mathbf{M}_1)$ *denote a set of generative factor pairs for* $\mathbf{M}_1$. *If the hidden size* $q = n_1 m_2 < \min(M, N)$, *then:*

$$\mathcal{F}_g(\mathbf{M}_1) \equiv \left\{ (\mathbf{L}, \mathbf{R}) \mid \mathbf{L} = \mathbf{L}_1 \boldsymbol{D},\ \mathbf{R} = \boldsymbol{P}_r^\top \boldsymbol{D}^{-1} \boldsymbol{P}_r \mathbf{R}_1; \boldsymbol{D} = \operatorname{diag}(\boldsymbol{v}) \in \mathcal{BD}, \boldsymbol{v} \in (\mathbb{R} \backslash \{0\})^q \right\}. \tag{9}$$

Theorem 3.4 shows that SaMA's invariance manifolds belong to the diagonal subgroups $\mathcal{BD}$. This is the consequence of the block-diagonal topology in $\mathbf{L}$ and $\mathbf{R}$ since dense rotations and shearing are not allowed. The proof can be found in Appendix D.6.2.

Implication. Pure scaling along coordinate axes neither mixes latent features nor disrupts Adam's axis-aligned preconditioner. Cross-feature entanglement is intrinsically suppressed by design, not by external regularization. The block-diagonal structure thus acts as an *implicit structural preconditioner*. It eliminates channel-mixing components involved in pathological flat directions arising in dense reparameterization, thereby reshaping the loss landscape into a form that adaptive optimizers can traverse more efficiently. Further discussion of parameter-space degeneracy is provided in Appendix D.6.

3.4.1 Empirical verification of gauge structure.

To empirically verify the theoretical distinction between LoRA and Morpho, we conduct toy experiments on a matrix approximation task.

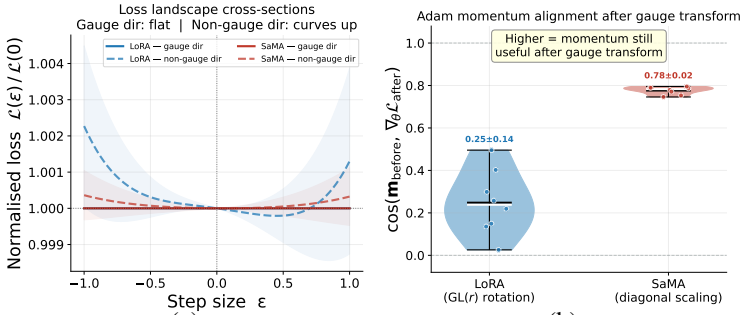

Figure 3: (a) Loss landscape flatness between LoRA and SaMA. (b) Cosine similarity between pre-transform Adam momentum $\boldsymbol{m}_{\text{before}}$ and post-transform gradient $\nabla_\theta \mathcal{L}_{\text{after}}$.

Loss landscape flatness. We partially trained both methods and scanned the normalized loss $\mathcal{L}(\theta + \varepsilon \mathbf{d}) \,/\, \mathcal{L}(\theta)$ along two directions: a gauge direction $\mathbf{d}_{\text{gauge}}$, which leaves $\Delta\mathbf{W}$ exactly unchanged, and a random non-gauge direction. As shown in Fig. 3(a), both gauge directions produce exactly flat loss curves (normalized loss $\equiv 1.0$), confirming the theoretical flat directions. Crucially, the non-gauge direction of Morpho remains nearly flat whereas LoRA's non-gauge direction exhibits significantly larger curvature. This indicates that the surrounding loss landscape of Morpho is easy to navigate.

Momentum alignment after gauge transform. To directly measure the impact of gauge redundancy on adaptive optimizers, we train each adapter for 400 warm-up steps with Adam to build meaningful first-moment estimates $\mathbf{m}$. We then apply a gauge transform that leaves $\Delta\mathbf{W}$ unchanged but modifies the parameter coordinates: a random permutation matrix $\mathbf{C}$ for LoRA, and an alternating diagonal scaling for Morpho. We then measure the cosine similarity between the pre-transform momentum

$\mathbf{m}_{\text{before}}$ and the post-transform gradient $\nabla_\theta \mathcal{L}_{\text{after}}$, which quantifies how much of Adam's accumulated directional information remains useful after the parameter coordinates are rearranged.

As shown in Fig. 3(b), LoRA yields a cosine similarity of 0.25 ± 0.14, indicating that the rotational freedom in $\mathrm{GL}(r)$ effectively scrambles Adam's momentum into an uninformative direction. In contrast, Morpho achieves 0.78 ± 0.02 with near-zero variance across seeds, confirming that diagonal scalings merely rescale the per-coordinate momentum without mixing features, leaving the optimizer state largely intact. These results provide direct empirical support for the claim that the $\mathrm{GL}(r)$ gauge freedom of LoRA creates an optimization bottleneck that Morpho's diagonal gauge structure inherently avoids.

4 Experimental Results

To evaluate SaMA against other PEFT methods, we conduct experiments across a broad spectrum of downstream tasks. Specifically, we verify SaMA effectiveness on three *complex reasoning* domains, namely DROP, Commonsense (CMS), and Arithmetic. Additional experiments, ablation studies, details of the datasets and hyperparameters are shown in Appendices F, G, I.1 and I.2, respectively.

Main baselines: Across these experiments, we compare SaMA against a range of strong recent PEFT baselines, including LoRA [7], PiSSA [8], ABBA [35], QuanTA [11], LoRA-Pro [36] and SMT [29], with full fine-tuning (FFT) included as a reference.

4.1 DROP Dataset

Following QuanTA [11], in order to explicitly clarify the effectiveness of these adapters, we conduct our initial experiments using Llama2-7B on the DROP dataset [37] as a representative case study for high-rank updates. As illustrated in Fig. 4 and further detailed in Table 9, SaMA consistently outperforms other existing fine-tuning methods across these models, reaching FFT (on Query and Value modules) performance. Notably, while increasing the rank to improve capacity, LoRA exhibits diminishing returns, failing to close the performance gap. Meanwhile, QuanTA – with high-rank updates – shows overall improvement, but still suffers from a performance lag.

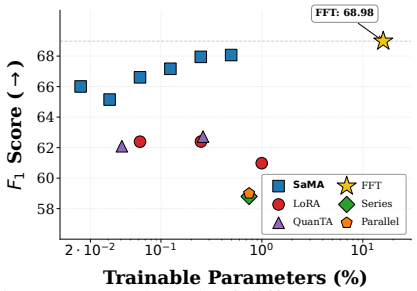

Figure 4: Benchmark of different FT methods on the DROP dataset, using Llama2-7B, as parameter budgets vary.

These results suggest that SaMA benefits from a mechanism that better navigates mid- and high-rank regimes. By leveraging sparse structures, SaMA facilitates a favorable optimization landscape, enabling a smoother optimization trajectory compared to dense high-rank alternatives. The omni-rank property might excel at extracting necessary features with less redundancy.

4.2 Arithmetic reasoning

Table 1: Comparison of PEFT methods across math benchmarks. Baselines (†) are from [35]. See Table 10 for the detailed report.

Method	Mistral-7B		Gemma-2 9B	
	GSM8K	MATH	GSM8K	MATH
Full FT†	63.87	17.65	79.23	38.02
$\text{LoRA}_{r=32}$	62.93	14.14	77.45	35.05
PiSSA†	62.43	16.52	77.12	37.04
LoRA-Pro†	63.07	17.32	78.26	37.53
$\text{ABBA}_{r=32}$	64.48	17.95	78.85	*39.59*
SaMA$_{8,8}$	*67.70*	*18.44*	*80.36*	39.36

To examine whether SaMA remains effective on demanding reasoning, we consider and evaluate models on MATH [38] and GSM8K [39] as representative challenging benchmarks among popular arithmetic reasoning tasks. We fine-tune pretrained models of Mistral-7B [40] and Gemma-2 9B [41] on a 20K-sample subset of MetaMathQA [42].

Results on the two test sets in Table 1 show a clear overview. Although all PEFT baselines adopt rank 32, equivalent to SaMA$_{32,32}$, SaMA consistently outperforms prior methods on both tasks while using fewer parameters; notably, even the lightweight setting $(n_1, m_2) = (8, 8)$ uses 4 times fewer parameters than the baselines, yet remains highly competitive and approaches FFT. This suggests that the proposed structured parameterization provides a favorable balance between efficiency and expressivity. We hypothesize that this effect arises by means of the structured parameterization, combined with a smoother optimization landscape.

4.3 Commonsense reasoning

Table 2: Performance comparison on commonsense reasoning tasks. The detailed benchmark with more FT baselines are provided in Table 11.

Model	PEFT method	#Params%	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	AVG(↑)
LLaMA-7B	FFT♠	100	70.3	84.2	80.1	92.3	85.4	86.6	72.8	83.4	81.9
	LoRA♠	0.83	69.2	81.7	78.4	83.4	80.8	79.0	62.4	78.4	76.7
	SMT•	0.84	68.7	81.7	78.3	91.6	78.8	84.1	68.7	77.4	78.7
	SaMA$_{16\times16}$	0.416	71.1	84.9	81.7	94.1	84.0	87.6	72.7	86.2	**82.8**
LLaMA-13B	FFT♠	100	74.5	86.3	81.3	94.4	86.9	89.7	77.9	88.8	85.0
	LoRA*	0.67	72.1	83.5	80.5	90.5	83.7	82.8	68.3	82.4	80.5
	SMT•	0.68	71.1	84.4	81.7	93.7	83.2	86.7	73.7	85.2	82.4
	SaMA$_{16\times16}$	0.336	73.5	87.6	82.6	95.4	87.7	90.2	78.0	88.3	85.4
LLaMA2-7B	FFT♠	100	74.7	84.9	78.7	93.7	84.1	87.5	75.2	85.0	83.0
	LoRA°	0.83	69.8	79.9	79.5	83.6	82.6	79.8	64.7	81.0	77.6
	SMT•	0.84	72.0	83.8	80.8	93.3	82.8	86.7	74.0	81.0	81.8
	SaMA$_{4\times4}$	0.104	73.1	85.5	81.8	94.7	85.0	88.7	73.9	86.0	83.6
	SaMA$_{16\times16}$	0.42	73.9	86.2	81.9	94.9	86.4	89.1	77.0	86.8	**84.5**
LLaMA3-8B	FFT♠	100	73.9	86.2	79.1	93.1	85.8	88.1	78.2	84.0	83.6
	LoRA°	0.70	70.8	85.2	79.9	91.7	84.3	84.2	71.2	79.0	80.8
	SMT•	0.71	75.7	88.4	81.4	96.2	88.2	92.7	83.2	88.6	86.8
	SaMA$_{4\times4}$	0.088	73.9	90.6	82.8	96.6	88.2	93.7	83.6	89.6	87.4
	SaMA$_{16\times16}$	0.351	75.4	90.3	83.2	96.7	88.7	94.1	83.8	90.1	**87.8**

We continue our evaluation on Commonsense reasoning tasks, including 8 benchmark datasets, training on a single amalgamated dataset of 147k samples from [43][1]. These experiments use several models in the Llama family [44, 45, 46] to make a comprehensive benchmark. Table 2 shows SaMA performance among other PEFT methods. In most of the models and sub-tasks, SaMA consistently yields significant improvements, sometimes surpasses FFT. This suggests that by leveraging sparse matrices, SaMA facilitates a favorable optimization landscape, allowing smoother optimization trajectories. Detailed results are provided in Table 11.

4.4 Training efficiency

We validate computational and memory efficiency of SaMA during the training phase by tracking time per step and reserved memory during the training time on Gemma2-9B (See details and additional results in Appendix F.6). The results are recorded after 50 warmup steps, averaged 3 times over 25 iterations with comparable parameter counts.

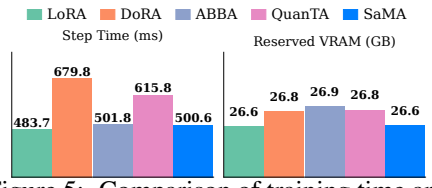

Figure 5: Comparison of training time and memory on Gemma2-9B. SaMA remains hardware-friendly, comparable to LoRA.

As shown in Fig. 5 and detailed in Appendix F.6, SaMA preserves hardware-friendly training behavior: it incurs only marginal runtime overhead of 3.5% and a comparable memory footprint close to LoRA, and noticeably lower cost than other dense baselines such as DoRA, QuanTA.

Summary Across diverse and challenging NLP tasks and models, SaMA consistently achieves competitive or superior performance while maintaining all facets of parameter efficiency, scaling stability, computational and memory effectiveness.

4.5 Discussion on disentangling the SaMA individual effects

Given the key paragraphs above, it is obvious to see that SaMA features several intrinsic crucial properties, namely Morpho structure, rank capacity as well as reduced gauge freedom of transforma-

[1] Refined from Commonsense-170K because of duplicates. See Section I.2.3

tion. Although SaMA adapters simultaneously offer expressivity, structure interaction and diagonal invariant manifolds, it naturally prompts a question on the root cause of SaMA performance:

What evidence helps disentangle the effects of these factors?

To thoroughly answer this thought-provoking question, we deem that absolutely isolating these inherently coupled components is mathematically challenging. However, their individual contributions can be observed through structured ablations, optimization dynamics and generalizability. Below is a brief discussion on this problem; the full analysis is provided in Appendix E.

i) Beyond nominal rank capacity. Although scaling rank consistently improves SaMA's expressivity across budgets (Fig. 19 (page 49)), rank capacity alone does not explain its superiority. In addition, SaMA preserves a rich, long-tailed singular value spectrum closely aligned with FFT (Fig. 23 page 52). See Appendix E.1.

ii) Isolating the Morpho topology (Global mixing). A naive diagonal or block-diagonal adapter such as DiaBlo [18] inherently possesses high rank and introduces no parameter degeneracy (since it does not contain any matrix factorization), yet restricts transformations to isolated local subspaces. Conversely, SaMA bridges distant feature dimensions to enable global information routing while scaling rank quadratically with budget, yielding better performance. See Appendix E.2.

iii) Isolating reduced gauge freedom (Invariant manifolds). Unlike LoRA's dense $\mathrm{GL}(r)$ invariance, which induces optimization concerns under Adam/AdamW (Section 3.4, page 5), SaMA's structure severely restricts gauge symmetry to scaling only, eliminating shearing and rotations. In our experiments, gauge-inflating SaMA by expanding the local gauge group to dense 16-by-16 blocks noticeably degrades performance on GSM8K and MATH (see Appendix E.3), providing evidence for the optimization benefit of reduced gauge freedom.

4.6 Ablation studies

Sensitivity analysis on initialization of SaMA factors

To assess the actual effect of zero-initialization of left and right factors in SaMA, we conduct experiments on the DROP dataset using Qwen2.5-1.5B [6], with several choices of standard deviation σ_{init} for normal distributions. All other training hyperparameters are fixed across the tests to ensure fair comparison, and the results are reported in Table 3 and Appendix F.7.

Table 3: Initialization strategies in SaMA.

$\sigma^2_{\mathbf{L}}$	$\sigma^2_{\mathbf{R}}$	Acc.	F_1-scores
0	1e−3	<u>36.04</u>	**39.77**
0	2e−4	**36.09**	<u>39.74</u>
1e−3	0	35.47	39.41
2e−4	0	35.29	39.13

In general, fine-tuning with Morpho parameterization is inherently *resilient* to zero-initialization. Moreover, it is more beneficial to initialize SaMA factors: $\mathbf{L} = 0$ and $\mathbf{R}$ from a zero-mean distribution with a small variance: $\mathbf{R} \sim \mathcal{N}(0, \sigma^2)$. Across both variance schemes of $\sigma^2_{\mathbf{L},\mathbf{R}}$, this configuration consistently improves overall performance relative to zero-$\mathbf{R}$ – for both accuracy and F_1-score metrics. Hence, we adopt this setting in all main experiments.

More results on ablation studies on fine-tuning performance on each individual transformer module and asymmetric configurations of (n_1, m_2) are provided in Appendix G.

5 Related Work

Parameter-Efficient Fine-Tuning (PEFT) PEFT methods broadly fall into three categories: addition, prompting, and reparameterization. The first two emerged in the early stage of LLM transfer learning, either inserting auxiliary modules [47, 48] or adapting via task-aware soft tokens [49, 50]. Both incur additional inference cost or sensitivity to initialization. Our work focuses on *reparameterization*, which preserves the prompt format and base architecture while introducing no inference overhead.

Reparameterization Reparameterization methods learn weight updates in a parameterized form and merge them back into the pretrained weights after fine-tuning. LoRA [7] and its variants [8,

28, 9, 10] have become the dominant baselines in this line, yet a clear performance gap to full fine-tuning (FFT) remains, motivating a recent push toward higher-rank updates. Representative high-rank approaches include QuanTA [11], which uses tensorization with multi-stage factorization; KRAdapter [12], based on the Khatri–Rao product; RandLoRA [51], which combines random bases; and SVD-based methods such as KASA [17], Spectral Adapter [13], and SVFT [52]. ABBA [35] also achieves rank expansion via Hadamard or Khatri–Rao products, but only up to $r^2/4$, and remains within the LoRA-style dense factorization that suffers from reparameterization ambiguity. In contrast, SaMA does not commit to either the low- or high-rank regime: a single two-factor parameterization spans both extremes with no preprocessing, while a structured sparse design avoids the dense-factor redundancy of LoRA-family extensions.

Structured sparse matrices Structured matrices admit subquadratic cost ($o(n^2)$ for $n \times n$ matrices) and include sparse families and fast transforms. Among them, butterfly [22, 53] and Monarch [21] matrices have seen renewed interest in deep learning, primarily for parameterizing *square* matrices in pre-training [54] or full fine-tuning [21]. SaMA instead extends these structures to a generic rectangular form and adapts them to additive PEFT, making structured sparsity scalable and rank-flexible for downstream fine-tuning.

6 Conclusion

In this study, we present SaMA together with the Morpho parameterization, which bridges the gulf between the Kronecker product and the Monarch matrices with practical applications in reparameterization fine-tuning. We support this framework with theory, mechanistic analyses and empirical validation. Our extensive experiments substantiate the effectiveness and applicability of the method across various tasks, model architectures and scales. SaMA consistently achieves competitive results against strong baselines. In particular, compared with FFT, SaMA substantially narrows the performance gap and is on par with that of FFT, while demonstrating high hardware friendliness and parameter efficiency. Concurrently, the precise conditions under which each rank regime is most beneficial and how far omni-rank PEFT can systematically close the gap to FFT in many settings remain open questions to be better understood.

More broadly, SaMA suggests that PEFT need not be confined to a fixed rank regime. SaMA leverages the sparse matrix structure to realize omni-rank adaptation within a unified framework. SaMA offers a broader view of PEFT methods beyond regime-specific adapter design. Our findings further suggest that the advantages arise not only from richer expressiveness, but also from a more favorable optimization geometry, without entanglement, under invariant manifolds.

Limitations SaMA implicitly relies on a specific structured parameterization, whose inductive bias may not be equally suitable across tasks, layers, and model families.

Future work There are several directions for future work to explore: **(i)** combining SaMA with other adaptation strategies, including ensemble-style integration or mixture-of-experts architectures; **(ii)** The core mechanism and underlying role of the sparse structure are not clear so far – how sparse matrix interactions can improve model generalization and enable the learning of a new task? **(iii)** Extension of the generic rectangular Morpho parameterization beyond fine-tuning, including its applicability to pretraining settings where many weight matrices are no longer square.[2]

[2]Monarch matrices mostly concentrate on square matrices and blocks.

Acknowledgements

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2019-II190079, Artificial Intelligence Graduate School Program (Korea University)), the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2025-23525509), and the High-Performance Computing Support Project, funded by the Government of the Republic of Korea (Ministry of Science and ICT) (No. RQT-25-070263).

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[2] Alexey Dosovitskiy, Lucas Beyer, Neil Houlsby, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*, 2020.

[3] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, et al. Gemini: A Family of Highly Capable Multimodal Models, 2025.

[4] Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, A. J. Ostrow, Akhila Ananthram, et al. OpenAI GPT-5 System Card, 2025.

[5] DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, et al. DeepSeek-V3.2: Pushing the Frontier of Open Large Language Models, 2025.

[6] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[7] Edward J Hu, Yelong Shen, Phillip Wallis, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *International Conference on Learning Representations (ICLR)*, 2022.

[8] Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems*, 2024.

[9] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. LoRA+: Efficient Low Rank Adaptation of Large Models. In *Forty-First International Conference on Machine Learning*, 2024.

[10] Tuan Truong, Chau Nguyen, Huy Nguyen, Minh Le, Trung Le, and Nhat Ho. RepLoRA: Reparameterizing low-rank adaptation via the perspective of mixture of experts. In *Proceedings of the 42nd International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2025.

[11] Zhuo Chen, Rumen Dangovski, Charlotte Loh, Owen Dugan, Di Luo, and Marin Soljačić. QuanTA: efficient high-rank fine-tuning of LLMs with quantum-informed tensor adaptation. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, 2024.

[12] Paul Albert, Frederic Z. Zhang, Hemanth Saratchandran, Anton van den Hengel, and Ehsan Abbasnejad. Towards Higher Effective Rank in Parameter-Efficient Fine-tuning using Khatri-Rao Product. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

[13] Fangzhao Zhang and Mert Pilanci. Spectral Adapter: Fine-Tuning in Spectral Space. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference for Learning Representations*, 2015.

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

[16] Dan Biderman, Jacob Portes Cody Blakeney, John P. Cunningham, et al. Lora learns less and forgets less. *Transactions on Machine Learning Research (TMLR)*, 2024.

[17] Fan Wang, Juyong Jiang, Chansung Park, Sunghun Kim, and Jing Tang. KaSA: Knowledge-Aware Singular-Value Adaptation of Large Language Models. In *The Thirteenth International Conference on Learning Representations*, 2024.

[18] Selcuk Gurses, Aozhong Zhang, Penghang Yin, Zi Yang, et al. DiaBlo: Diagonal Blocks Are Sufficient For Finetuning. In *The Fourteenth International Conference on Learning Representations*, 2025.

[19] Qiushi Huang, Tom Ko, Zhan Zhuang, Lilian Tang, and Yu Zhang. HiRA: Parameter-efficient hadamard high-rank adaptation for large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[20] Harold V. Henderson, Friedrich Pukelsheim, and Shayle R. Searle. On the history of the kronecker product. *Linear and Multilinear Algebra*, 1983.

[21] Tri Dao, Beidi Chen, Nimit S. Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Re. Monarch: Expressive Structured Matrices for Efficient and Accurate Training. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

[22] Tri Dao, Albert Gu, Matthew Eichhorn, Atri Rudra, and Christopher Re. Learning Fast Algorithms for Linear Transforms Using Butterfly Factorizations. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[23] David H. Bailey. Ffts in external or hierarchical memory. *The Journal of Supercomputing*, 1990.

[24] Charles F. Van Loan. The ubiquitous Kronecker product. *Journal of Computational and Applied Mathematics*, 2000.

[25] Huamin Zhang and Feng Ding. On the Kronecker products and their applications. *Journal of Applied Mathematics*, 2013.

[26] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In *Proceedings of 6th International Conference on Learning Representations, ICLR*, 2018.

[27] Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of ACL-IJCNLP*. ACL, 2021.

[28] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[29] Haoze He, Juncheng Li, Xuan Jiang, and Heather Miller. SMT: Fine-tuning large language models with sparse matrices. In *International Conference on Learning Representations (ICLR)*, 2025.

[30] Wenxuan Tan, Nicholas Roberts, Tzu-Heng Huang, Jitian Zhao, John Cooper, Samuel Guo, Chengyu Duan, and Frederic Sala. MoRe Fine-Tuning with 10x Fewer Parameters, 2024.

[31] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1936.

[32] Olivier Roy and Martin Vetterli. The Effective Rank: A Measure of Effective Dimensionality. In *2007 15th European Signal Processing Conference*, 2007.

[33] Damjan Kalajdzievski. A rank stabilization scaling factor for fine-tuning with LoRA. *ArXiv*, abs/2312.03732, 2023.

[34] Adela DePavia, Vasileios Charisopoulos, and Rebecca Willett. How do simple rotations affect the implicit bias of Adam?, 2025.

[35] Raghav Singhal, Kaustubh Ponkshe, Rohit Vartak, and Praneeth Vepakomma. Abba: Highly expressive hadamard product adaptation for large language models. *International Conference on Learning Representations (ICLR)*, 2026.

[36] Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. LoRA-Pro: Are Low-Rank Adapters Properly Optimized? In *The Thirteenth International Conference on Learning Representations*, 2025.

[37] Dheeru Dua, Yizhong Wang, Matt Gardner, et al. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. ACL, 2019.

[38] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring Mathematical Problem Solving With the MATH Dataset. In *Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[39] Karl Cobbe, Vineet Kosaraju, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[40] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, et al. Mistral 7b, 2023.

[41] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, et al. Gemma 2: Improving open language models at a practical size, 2024.

[42] Longhui Yu, Weisen Jiang, et al. Metamath: Bootstrap your own mathematical questions for large language models. *International Conference on Learning Representations (ICLR)*, 2024.

[43] Zhiqiang Hu, Lei Wang, Soujanya Poria, Roy Lee, et al. LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2023.

[44] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, et al. Llama: Open and efficient foundation language models. *CoRR*, 2023.

[45] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, et al. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288, 2023.

[46] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, et al. The llama 3 herd of models. arXiv preprint arXiv:2407.21783, 2024.

[47] Neil Houlsby, Andrei Giurgiu, Mona Attariyan, Sylvain Gelly, et al. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[48] Shwai He, Liang Ding, Daize Dong, Jeremy Zhang, and Dacheng Tao. SparseAdapter: An Easy Approach for Improving the Parameter-Efficiency of Adapters. In *Findings of the Association for Computational Linguistics: EMNLP 2022*. Association for Computational Linguistics, 2022.

[49] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 2021.

[50] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2021.

[51] Paul Albert, Frederic Z Zhang, Cristian Rodriguez-Opazo, Hemanth Saratchandran, Anton van den Hengel, and Ehsan Abbasnejad. Randlora: full rank parameter-efficient fine-tuning of large models. *International Conference on Learning Representations (ICLR)*, 2025.

[52] Vijay Lingam, Atula Tejaswi Neerkaje, Sujay Sanghavi, et al. SVFT: Parameter-Efficient Fine-Tuning with Singular Vectors. In *The 38th Annual Conference on Neural Information Processing Systems*, 2024.

[53] D.S. Parker. *Random Butterfly Transformations with Applications in Computational Linear Algebra*. CSD (Series). UCLA Computer Science Department, 1995.

[54] Anna Thomas, Albert Gu, Tri Dao, Atri Rudra, and Christopher Ré. Learning Compressed Transforms with Low Displacement Rank. In *Advances in Neural Information Processing Systems*, 2018.

[55] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*. ACL, 2021.

[56] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a Unified View of Parameter-Efficient Transfer Learning. In *International Conference on Learning Representations*, 2021.

[57] Guanghui Qin and Jason Eisner. Learning How to Ask: Querying LMs with Mixtures of Soft Prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2021.

[58] Yihan Wang, Si Si, Daliang Li, Michal Lukasik, Felix Yu, Cho-Jui Hsieh, Inderjit S. Dhillon, and Sanjiv Kumar. Two-stage LLM Fine-tuning with Less Specialization and More Generalization. In *The Twelfth International Conference on Learning Representations*, 2023.

[59] Zeju Qiu, Weiyang Liu, Haiwen Feng, Yuxuan Xue, Yao Feng, Zhen Liu, Dan Zhang, Adrian Weller, and Bernhard Schölkopf. Controlling Text-to-Image Diffusion by Orthogonal Finetuning. In *Thirty-Seventh Conference on Neural Information Processing Systems*, 2023.

[60] Mikhail Gorbunov, Kolya Yudin, Vera Soboleva, Aibek Alanov, Alexey Naumov, and Maxim Rakhuba. Group and Shuffle: Efficient Structured Orthogonal Parametrization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[61] An Nguyen, Jaesik Choi, and Anh Tong. LoCO: Low-Rank Compositional Rotation Fine-tuning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2026.

[62] Xinyu Ma, Xu Chu, Zhibang Yang, Yang Lin, Xin Gao, and Junfeng Zhao. Parameter Efficient Quasi-Orthogonal Fine-Tuning via Givens Rotation. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[63] Qingru Zhang, Minshuo Chen, Weizhu Chen, Tuo Zhao, et al. Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. In *The Eleventh International Conference on Learning Representations*, 2022.

[64] Hanqing Wang, Yixia Li, Shuo Wang, Guanhua Chen, and Yun Chen. Milora: Harnessing minor singular components for parameter-efficient llm finetuning. *arXiv preprint arXiv:2406.09044*, 2024.

[65] Muqing Liu, Chongjie Si, and Yuheng Jia. FlexLoRA: Entropy-Guided Flexible Low-Rank Adaptation. In *The Fourteenth International Conference on Learning Representations*, 2025.

[66] Massimo Bini, Leander Girrbach, and Zeynep Akata. Decoupling Angles and Strength in Low-rank Adaptation. In *The Thirteenth International Conference on Learning Representations*, 2024.

[67] Kai Lion, Liang Zhang, Bingcong Li, and Niao He. PoLAR: Polar-Decomposed Low-Rank Adapter Representation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[68] Soroush Abbasi Koohpayegani, Hamed Pirsiavash, et al. NOLA: Compressing LoRA using Linear Combination of Random Basis. In *International Conference on Learning Representations*, 2023.

[69] Zhiwei He, Zhaopeng Tu, Xing Wang, Rui Wang, et al. RaSA: Rank-Sharing Low-Rank Adaptation. In *The Thirteenth International Conference on Learning Representations*, 2024.

[70] Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. VeRA: Vector-based Random Matrix Adaptation. In *The Twelfth International Conference on Learning Representations*, 2023.

[71] Yong Liu, Di Fu, Shenggan Cheng, Zirui Zhu, Yang Luo, Yang You, et al. SeedLoRA: A Fusion Approach to Efficient LLM Fine-Tuning. In *Forty-Second International Conference on Machine Learning*, 2025.

[72] Yilang Zhang, Bingcong Li, and Georgios B. Giannakis. RefLoRA: Refactored Low-Rank Adaptation for Efficient Fine-Tuning of Large Models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[73] Shaowen Wang, Linxi Yu, and Jian Li. LoRA-GA: Low-Rank Adaptation with Gradient Approximation. *Advances in Neural Information Processing Systems*, 37, 2024.

[74] Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection. In *Forty-First International Conference on Machine Learning*, 2024.

[75] Yuxi Liu, Renjia Deng, Kun Yuan, et al. MISA: Memory-Efficient LLMs Optimization with Module-wise Importance Sampling. In *The Thirty-ninth Conference on Neural Information Processing Systems*, 2025.

[76] Zicheng Zhang, Haoran Li, Yifeng Zhang, Guoqiang Gong, Jiaxing Wang, Junxing Hu, Pengzhang Liu, and Qixia Jiang. The Primacy of Magnitude in Low-Rank Adaptation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[77] Anh Tong, Kyowoon Lee, and Jaesik Choi. PRISM: Principal Subspace Alignment for Parameter-Efficient Fine-Tuning. In *Advances in Neural Information Processing Systems*, 2026.

[78] Yeonjoon Jung, Daehyun Ahn, Hyungjun Kim, Taesu Kim, and Eunhyeok Park. GraLoRA: Granular Low-Rank Adaptation for Parameter-Efficient Fine-Tuning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[79] Weixi Song, Zuchao Li, Lefei Zhang, Hai Zhao, and Bo Du. Sparse is Enough in Fine-tuning Pre-trained Large Language Models. In *Forty-First International Conference on Machine Learning*, 2024.

[80] Jing Xu and Jingzhao Zhang. Random Masking Finds Winning Tickets for Parameter Efficient Fine-tuning. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[81] Rui Pan, Xiang Liu, Shizhe Diao, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. LISA: Layerwise Importance Sampling for Memory-Efficient Large Language Model Fine-Tuning. *Advances in Neural Information Processing Systems*, 37, 2024.

[82] Xinyu Yang, Huaxiu Leng, Jixuan Yao, Beidi Chen, et al. S^2FT: Efficient, Scalable and Generalizable LLM Fine-tuning by Structured Sparsity. *Advances in Neural Information Processing Systems*, 2024.

[83] Tri Dao, Nimit Sohoni, Albert Gu, Matthew Eichhorn, Amit Blonder, Megan Leszczynski, Atri Rudra, and Christopher Ré. Kaleidoscope: An Efficient, Learnable Representation For All Structured Linear Maps. In *International Conference on Learning Representations*, 2019.

[84] Beidi Chen, Tri Dao, Kaizhao Liang, Jiaming Yang, Zhao Song, Atri Rudra, and Christopher Re. Pixelated Butterfly: Simple and Efficient Sparse training for Neural Network Models. In *International Conference on Learning Representations*, 2021.

[85] Behnam Neyshabur, Russ R Salakhutdinov, and Nati Srebro. Path-SGD: Path-Normalized Optimization in Deep Neural Networks. In *Advances in Neural Information Processing Systems*, 2015.

[86] Fangzhao Zhang and Mert Pilanci. Riemannian Preconditioned LoRA for Fine-Tuning Foundation Models. In *Forty-First International Conference on Machine Learning*, 2024.

[87] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 2012.

[88] James Joseph Sylvester. On involutants and other allied species of invariants to matrix systems. *Johns Hopkins University Circulars*, 3:9–12, 34–35, 1884.

[89] Haskell P. Rosenthal. On the subspaces oflp(p>2) spanned by sequences of independent random variables. *Israel Journal of Mathematics*, 1970.

[90] V A Marčenko and L A Pastur. Distribution of eigenvalues for some sets of random matrices. *Mathematics of the USSR-Sbornik*, 1967.

[91] Zhidong D Bai and Yanqing Yin. Necessary and Sufficient Conditions for Almost Sure Convergence of the Largest Eigenvalue of a Wigner Matrix. *The Annals of Probability*, 1988.

[92] Iain M. Johnstone. On the distribution of the largest eigenvalue in principal components analysis. *The Annals of Statistics*, 29(2), 2001.

[93] Pratyusha Sharma, Jordan T. Ash, and Dipendra Misra. The Truth is in There: Improving Reasoning in Language Models with Layer-Selective Rank Reduction. In *The Twelfth International Conference on Learning Representations*, 2023.

[94] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 4th edition, 2013.

[95] T. N. E. Greville. Note on the generalized inverse of a matrix product. *SIAM Rev.*, 1966.

[96] Weiyang Liu, Zeju Qiu, Yao Feng, Yuliang Xiu, Yuxuan Xue, Longhui Yu, Haiwen Feng, Zhen Liu, Juyeon Heo, Songyou Peng, et al. Parameter-efficient orthogonal finetuning via butterfly factorization. In *The Twelfth International Conference on Learning Representations*, 2024.

[97] Roger Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(3):406–413, 1955.

[98] Ruijie Zhang, Ziyue Liu, Zhengyang Wang, and Zheng Zhang. LaX: Boosting Low-Rank Training of Foundation Models via Latent Crossing. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[99] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[100] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, et al. A large-scale study of representation learning with the visual task adaptation benchmark, 2020.

[101] Wei Dong, Xing Zhang, Yang Yang, et al. Low-Rank Rescaled Vision Transformer Fine-Tuning: A Residual Design Approach. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[102] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer Feed-Forward Layers Are Key-Value Memories. In *Conference on Empirical Methods in Natural Language Processing*, 2021.

[103] Sepp Hochreiter and Jürgen Schmidhuber. Flat Minima. *Neural Computation*, 1997.

[104] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp Minima Can Generalize For Deep Nets. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2017.

[105] Nitish Shirish Keskar, Ping Tak Peter Tang, et al. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In *International Conference on Learning Representations*, 2017.

[106] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware Minimization for Efficiently Improving Generalization. In *International Conference on Learning Representations*, 2020.

[107] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

[108] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *AAAI Conference on Artificial Intelligence*, 2019.

[109] Maarten Sap, Hannah Rashkin, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. In *Conference on Empirical Methods in Natural Language Processing*, 2019.

[110] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[111] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: an adversarial winograd schema challenge at scale. *Communications of the ACM*, 2021.

[112] Hector J. Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *The 13rd International Conference on Principles of Knowledge Representation and Reasoning*, 2012.

[113] Peter Clark, Isaac Cowhey, Oren Etzioni, et al. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

[114] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. ACL, 2018.

[115] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.

Table of Appendix Contents

A Mathematical Notations **18**

B Broader Literature Review **19**

C Practical Implementation **20**

D Extended Theoretical Discussion and Proofs **22**

D.1 Rank property (Proof of Theorem 3.1) 22

D.2 Rank stability condition (Proof of Theorem 3.3) 23

D.3 Expressivity in low effective rank. Proof of Theorem 3.2 26

D.4 Numerical stability of SaMA over ranks 31

D.5 SaMA Converges to the FFT Solution 32

D.6 Gauge Invariance in PEFT: A Unified Perspective and the Parameter-Space Uniqueness of Morpho 35

D.7 Distinction from MoRE Fine-tuning 37

E What evidence supports disentangling the effects of SaMA adapter features **41**

E.1 Expressivity via rank capacity 41

E.2 Morpho structure 41

E.3 Invariant manifolds 42

F Further Experimental Result and Analysis **43**

F.1 Matrix approximation – experimental settings 43

F.2 Full results on the DROP dataset 44

F.3 Full results on the math reasoning task 44

F.4 A comprehensive benchmark of Commonsense reasoning 45

F.5 Attention fine-tuning on Commonsense reasoning 47

F.6 Efficiency validation 47

F.7 Ablation study on adapter initialization 48

F.8 SaMA performance when increasing parameter budgets 49

F.9 Fine-tuning Vision transformer 49

G Ablation studies **51**

G.1 Module-specific adaptation 51

G.2 Asymmetric configuration of (n_1, m_2) 51

H Understanding the adapter via mechanistic analysis **51**

H.1 Does SaMA leverage the Omni-rank property? 51

H.2 Gradient behavior during optimization 52

I Dataset and experimental details **54**

I.1 Details of datasets on experiments 54

I.2 Experimental descriptions 55

I.3 Template of Instruction fine-tuning 59

J Broader Impact **61**

A Mathematical Notations

Frequently used notation: Table 4 summarizes the common symbols used throughout this work.

Table 4: General linear-algebra notation used throughout the paper.

Symbol	Meaning
$\mathbf{A}, \mathbf{B}, \mathbf{C}$	Generic matrices; in the Kronecker formulation, $\mathbf{A} \in \mathbb{R}^{m_1 \times n_1}$ and $\mathbf{B} \in \mathbb{R}^{m_2 \times n_2}$.
$\otimes$	Kronecker product.
$\mathbf{x}, \mathbf{y}$	Input and output vectors of a linear transformation.
$\mathbf{X}, \mathbf{Y}$	Matrix forms of $\mathbf{x}, \mathbf{y}$ obtained via $\mathrm{unvec}(\cdot)$.
$\mathrm{vec}(\cdot), \mathrm{unvec}(\cdot)$	Vectorization and inverse vectorization operators.
M, N	(Default) Output and input dimensions of the target linear map or an adapter matrix. For a square matrix, we denote its size as N or N-by-N.
m_1, n_1, m_2, n_2	Block dimensions and configuration parameters in the Kronecker/SaMA construction.
$\mathrm{rank}(\cdot)$	Rank operator.
$\mathrm{erank}(\cdot)$	Shannon entropy effective rank operator.
r	Rank hyperparameter in LoRA-style baselines.
$\mathcal{G}_X$	Gauge group associated with the invariance manifold of a generic parameterization X.
$GL(r)$	General linear group, i.e., the set of all invertible square matrices of size $r \times r$.
$O(r)$	Rotational group – a set of rotational square matrices. $\mathbf{A} \in O(r)$ means that $\mathbf{A}^{-1} = \mathbf{A}^\top$ and $\det(\mathbf{A}) = 1$.
$\mathcal{BD}$	Set of all block-diagonal matrices.
$\tilde{\mathbb{R}}$	$\tilde{\mathbb{R}} = \mathbb{R} \backslash \{0\}$: the set of nonzero real numbers.
$\|\cdot\|_F$	Frobenius norm of a matrix.
$\langle \cdot, \cdot \rangle_F$	Frobenius inner product of two matrices.
$\boldsymbol{e}_i$	Standard basis vector with only a 1 in the i-th position and zeros otherwise.
$\pi()$	Permutation function - a bijective function that rearranges a set of indices. For instance, $\pi(i) = j$ means that the j-th element is moved to the i-th position
$\pi_{m,n}$	Index map of $\boldsymbol{P}_{m,n}$, i.e. $\boldsymbol{P}_{m,n}\boldsymbol{e}_i = \boldsymbol{e}\pi_{m,n}(i)$
$\overline{a..b}$	Integer interval ranging from a to b – or $[a, b] \cap \mathbb{Z}$

SaMA-specific notation. Table 5 summarizes the symbols used for SaMA and Morpho matrices.

Table 5: SaMA-specific notation.

Symbol	Meaning
$\Delta\mathbf{W}$	Trainable adapter update added to the pretrained weight matrix in FT
$\mathbf{L}$, $\mathbf{R}$	Block-diagonal trainable factors in the Morpho parameterization, with constituent blocks as $\mathbf{L}_i$ and $\mathbf{R}_j$.
$\boldsymbol{P}$ or $\boldsymbol{P}_{n_1,m_2}$	A generic form of perfect shuffle permutation. In general, $\boldsymbol{P}_{m,n}$ rearranges a vector by reshaping it into a row-major m-by-n matrix then transposes and re-vectorizing it.
$\boldsymbol{P}_1$	Permutation matrix for monarch parameterization of all square components.
$\boldsymbol{P}_l, \boldsymbol{P}_r$	Fixed permutation matrices used to route and mix block interactions. In Morpho parameterization, $\boldsymbol{P}_l = \boldsymbol{P}_{m_1,m_2}, \boldsymbol{P}_r = \boldsymbol{P}_{n_1,m_2}$.
s, s_{SaMA}	Scaling factor applied to the SaMA adapter; for stable scaling, $s = \Theta(n_1^{1/4})$.
q	Hidden dimension of the Morpho parameterization, defined as $q = n_1 \cdot m_2$
$\mathcal{F}, \mathcal{F}^{(m,n)}$	Morpho manifolds with or without specific primary configuration, parameterized by (#blocks in $\mathbf{R}$, #blocks in $\mathbf{L}$).
$\mathcal{F}_g, \mathcal{F}_g^{(m,n)}$	Generative factor tuples $(\mathbf{L}, \mathbf{R})$ associated with the Morpho manifold.
(n_1, m_2)	Primary configuration of SaMA: (#blocks in $\mathbf{R}$, #blocks in $\mathbf{L}$).

We summarize the recurring symbols used across the main text and appendix. These are common quantities and operators that appear across multiple sections, although some may be redefined locally to make the context clear. Context-specific notation will be introduced in the corresponding sections. For the sake of readability, we separate general linear-algebra notation from SaMA-specific notation, as displayed in Table 4 and Table 5.

Definition A.1. *Perfect shuffle permutation matrix of shape $r \times r$ [24]:*

$$\boldsymbol{P}_{p,q} = \begin{bmatrix} \mathrm{I}_r[1:q:r,:]^\top & \mathrm{I}_r[2:q:r,:]^\top & \dots & \mathrm{I}_r[q:q:r,:]^\top \end{bmatrix}^\top, \quad r = pq. \tag{10}$$

As explained in the main text, the perfect shuffle permutation $\boldsymbol{P}_{p,q}$ reshapes a vector of length r into a $p \times q$ matrix in row-major order, transposes the matrix, then re-vectorizes it.

Terminology explanation *Omni-rank:* ability to realize a broad range of matrix ranks, up to the target regime controlled by the configuration, including low-, mid-, high-, and full-rank transformations. Within this work, it is also referred to as flexible rank or quadratic rank property.

In this work, we define *omni-rank* not as the empirical capacity to realize every possible integer rank, but rather as the structural flexibility to span a broad spectrum of adapter ranks while respecting the available parameter budget.

Morpho: structured parameterization underlying SaMA, built from sparse block-diagonal factors and the corresponding perfect shuffle permutations. It can be viewed as a natural extension of the Kronecker product.

B Broader Literature Review

Additional adapter: The first group can be referred to as Addition approaches, which involves injecting extra adapters into the model architecture. These modules can be integrated in many ways: Serial adapters are interposed between existing attention or multilayer perceptron (MLP) components [47, 55, 48], while parallel adapters coexist alongside these modules [56].

Prompting Secondly, the Prompting group: inspired by the success of textual prompting methods, Prompt tuning [50] and Prefix tuning [49] add extra soft tokens to the original input and solely focus on fine-tuning these trainable tokens, or soft prompts in [57]. Besides, a recent work by [58] proposes two-stage fine-tuning, where prompting techniques are used in the first stage. Overall, these methods commonly encounter efficacy issues due to their sensitivity to initialization affecting their overall robustness and efficiency. These first two categories, either altering the model's input sequence or architecture, lead to an increase in model size and introduce additional inference latency and computational overhead compared to the pretrained models. So the research directions were prevalent in the early time of PEFT, but less advanced recently, so they are outside of the scope of this paper.

Isometric-based Fine-tuning Grounded in the metric preservation insight from reparameterization fine-tuning, there are other directions that use orthogonal adapters in the form of multiplicative transformations. Particularly, the orthogonal matrices are parameterized via block-diagonal structures in OFT [59] or GS [60], which also utilize sparsity structures; Moreover, LoCO [61] constructs rotations from low-rank skew-symmetric matrices by means of Woodbury identity; some works focus on simple components like Givens rotations in qGOFT [62]. Our work, in contrast, takes additive transformations as a core principle because of several prominent advantages. We argue that additive adapters are more effective and expressive in PEFT due to the simplicity and mathematical generality .

Reparameterization *Low-rank Adaptation*. Building on the low intrinsic rank hypothesis [27], LoRA [7] uses a product of two low-rank matrices to build weight updates, avoiding inference overhead of previous approaches with adapters [47], or optimization bottleneck in prompt tuning[49, 50]; LoRA then quickly becomes one of the most common techniques. Beyond the vanilla design, subsequent works extend this framework through adaptive rank allocation [63], alternative initialization [8, 64], spectrum [65], decomposition [28, 66, 67], Hadamard product [19, 35], sharing components [68, 69], random non-trainable matrices [70, 68], fusion [71], training resources (gradient [36, 72, 73], memory [74, 75], learning dynamics [9, 33, 76]), attention interaction [77] or reparameterization [10, 78].

ABBA [35] also exhibits a rank-expansion effect, indirectly suggesting a limited form of *Omni-rank* – $r^2/4$ only. However, this property is not explicitly formulated as a central design principle or motivation in their work. In contrast, SaMA directly builds on the notion of *Omni-rank* adaptation as a primary objective, and develops both the methodology and analyses around this perspective. Moreover, ABBA remains fundamentally within the LoRA-style family, and thus still inherits the reparameterization ambiguity associated with dense factorized updates. Conversely, SaMA is designed to achieve the *Omni-rank* target of $n_1 \cdot m_2$, equivalent to r^2, while disentangling gauge symmetry (Sec. 3.4) and maintaining an implementation-friendly sparse architecture (Appendix C).

Partial specification and sparse updates This line of approach aims to reduce the trainable parameters by selecting a subset of pre-trained model weights or sparse matrices in additive adapters. This kind of method attracts much attention in recent research but either sacrifice hardware efficiency [29], even requiring full backward propagation [79, 80] imposing additional training procedures [81, 82], or only permits local interaction without dense matrices [18]. In addition, MoRE [30] can be conceptually viewed as an ensemble of b LoRA-style block adapters of rank r_{blk}, combined with permutation. This partitioned design restricts the global receptive field, leading to a fragmented interaction topology. Apart from those, our work facilitates sparse constituent matrices to construct module-specific dense adapters that are more practical, easy-to-implement while maintaining simplicity and efficiency by virtue of reparameterization for each individual weight matrix.

Structured sparse matrices Structured matrices are those with more optimal time complexity and space complexity – below the threshold of quadratic for a square matrix $n \times n$. Many generations of these transformations have been adopted in machine learning to alter dense weight matrices. In [83], the expressiveness of butterfly matrices [22] can cover these structured matrices almost optimally regarding runtime and memory, followed by the proposal of Pixelated Butterfly [84] to make them hardware-friendly. Moreover, Dao et al. [21] show that monarch representation is strictly more general than butterfly matrices while enhancing hardware utilization; the product of two monarch matrices can represent convolution, Hadamard transform, Toeplitz matrices and AFDF matrices.

Geometric perspectives and Manifold reparameterization in PEFT In deep learning, the structural formulation of models inherently introduces continuous gauge symmetries into the parameter space, as highlighted in [85]. In the context of PEFT, gauge symmetry is implicitly addressed via constraints proposed in recent works. For instance, Riemannian LoRA [86] handles the reparameterization issues by modeling the low-rank adaptation space strictly as a Riemannian submanifold. LoRA+ [9] reveals that parameter space degeneracy in LoRA induces asymmetric optimization dynamics, leading to suboptimal feature learning. Our work explicitly analyzes the gauge symmetry in the parameter space and resolves it through rigid structural constraints. By adopting sparse structured matrices, our approach directly breaks the transform structure of the continuous gauge invariance inherent in standard dense factorization; replacing them with block-diagonal matrices intrinsically manages parameter entanglement.

C Practical Implementation

This section provides the practical details of SaMA: the trainable parameter count of a Morpho adapter, the initialization scheme inherited from LoRA, the efficient batched block-matmul realization that avoids materializing $\Delta\mathbf{W}$, and the absence of both pre-processing and inference-time overhead.

Parameters. For a square weight matrix $\mathbf{W} \in \mathbb{R}^{N\times N}$ with $n_1 = m_2 = \nu$, a Morpho matrix $\mathbf{M} = \boldsymbol{P}_l^\top \mathbf{L} \boldsymbol{P}_r \mathbf{R}$ is described by $2N\nu$ trainable parameters in total: each of $\mathbf{L}$ and $\mathbf{R}$ contains ν dense blocks of size $\nu \times {}^{N}/_{\nu}$, contributing N parameters per factor. The permutations $\boldsymbol{P}_l, \boldsymbol{P}_r$ are fixed and parameter-free.

Initialization. Following the reparameterization convention of LoRA [7], we initialize $\mathbf{R}$ from a zero-mean Gaussian with small standard deviation $\sigma_{\mathbf{R}}$, and set $\mathbf{L} = 0$. This ensures $\Delta\mathbf{W} = s\, \boldsymbol{P}_l^\top \mathbf{L} \boldsymbol{P}_r \mathbf{R} = 0$ at the start of training while keeping gradients non-vanishing. The scalar s_{SaMA} is a configuration-dependent hyperparameter whose role in training stability is analyzed in Theorem 3.3.

Efficient computation. Although $\Delta\mathbf{W}$ in Eq. (4) defines a dense linear map, we never materialize it explicitly. Instead, following [21], multiplication $\Delta\mathbf{W}\,\mathbf{x}$ is carried out in two steps: reshape the input into a matrix, apply $\mathbf{R}$ as a batched block-matrix multiply, permute via $\boldsymbol{P}_r$, apply $\mathbf{L}$ similarly, then

permute via $\boldsymbol{P}_l^\top$. This exploits batch matrix multiplication — the core primitive of modern GPU hardware — at a total cost of $O(\nu N)$ FLOPs, with no recursive graph construction [22].

No pre-processing. Unlike D2S-FT [21], which requires projecting pre-trained weights into a sparse form before training begins, SaMA requires no such initialization step. Pre-trained weights are frozen and never modified; all computation targets the sparse adapter $\Delta\mathbf{W}$ alone. This eliminates the Dense-to-Sparse projection overhead present in related methods [81, 64] while preserving comparable operational complexity.

No inference overhead. Like other reparameterization-based methods, the SaMA adapter can be merged into the pre-trained weight at inference time ($\mathbf{W}_{\text{ft}} = \mathbf{W}_0 + \Delta\mathbf{W}$), adding zero latency and preserving the original model architecture for optimized deployment. Only the lightweight adapter needs to be stored, not the full fine-tuned model.

D Extended Theoretical Discussion and Proofs

D.1 Rank property (Proof of Theorem 3.1)

Theorem. *Let* $\mathbf{L}$ *and* $\mathbf{R}$ *be two block-diagonal matrices, as defined in Eq. (5), with* $n_1 m_2 \leq \min(M, N)$. *Suppose that within every submatrix, each entry is chosen from independent and identically distributed data (i.i.d). Then, the rank of the SaMA adapter matrix:*

$$\operatorname{rank}(\Delta\mathbf{W}) = \operatorname{rank}(s\boldsymbol{P}_l^\top \mathbf{L}\boldsymbol{P}_r\mathbf{R}) = n_1 m_2, \tag{11}$$

holds with a probability of 1.

Theorem 3.1 characterizes the rank that SaMA can attain during training; it does not describe the initialization, where $\mathbf{L} = \mathbf{0}$ and hence $\Delta\mathbf{W} = \mathbf{0}$. Crucially, this theorem highlights a significant distinction from low-rank adaptation – the LoRA family [64, 8], and high/full-rank adaptation [11, 12, 19]. Whereas LoRA [7] utilizes the *'low intrinsic rank'* assumption [27], SaMA further leverages the principle with regarding to the current development of large models and complex downstream tasks.

Lemma D.1. *Let* $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_{m_2}$ *denote* m_2 *vectors in* $\mathbb{R}^{n_2}$, *where* $m_2 \leq n_2$, *drawn i.i.d from a Gaussian or any other distribution. Then with probability 1, the set* $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_{m_2}$ *will be linearly independent.*

Proof of the lemma. We first note that any measure defined via a Gaussian or any other continuous probability distribution is absolutely continuous with respect to the Lebesgue measure. This implies that such distributions assign zero probability to any set of Lebesgue measure zero.

Consider the inductive process of drawing the set $\{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_{m_2}\}$ in $\mathbb{R}^{n_2}$. Since the vectors are drawn independently, we first consider $\mathbf{X}_1$. The probability that $\mathbf{X}_1$ is the zero vector is 0 with respect to the Lebesgue measure on $\mathbb{R}^{n_2}$, and consequently for any measure absolutely continuous to it. Upon drawing the subsequent vector $\mathbf{X}_{k+1}$ (where $k < m_2 \leq n_2$), the probability that $\mathbf{X}_{k+1}$ lies in the subspace $V_k = \operatorname{span}\{\mathbf{X}_1, \ldots, \mathbf{X}_k\}$ is also 0. This is because V_k is a subspace of dimension at most k, which constitutes a set of zero Lebesgue measure in $\mathbb{R}^{n_2}$.

By induction, it follows that the set $\{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_{m_2}\}$ is linearly independent with probability 1 (almost surely). For the general case where vectors may not be restricted in length, we note that normalizing $\mathbf{X}_i$ by its non-zero norm does not affect linear independence; thus, the result holds for any continuous distribution on $\mathbb{R}^{n_2}$. □

Proof of Theorem 3.1

Main proof. Consider a block-diagonal matrix $\mathbf{R} = \operatorname{diag}(\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_{n_1})$, where $\mathbf{R}_i \in \mathbb{R}^{m_2 \times n_2}$. For any pairs of $(\mathbf{R}_i, \mathbf{R}_j), i \neq j$, their columns and rows are in different locations in $\mathbf{R}$. Due to sparsity of $\mathbf{R}$, the blocks $\{\mathbf{R}_1, \ldots, \mathbf{R}_{n-1}\}$ occupy pairwise disjoint set of rows/columns; hence it is the case that these columns/rows in the first submatrix are not a linear combination of columns/rows from the other submatrices and vice versa. It follows that the overall rank of $\mathbf{R}$ is simply a summation over the constituent block elements $\mathbf{R}_i$.

$$\operatorname{rank}(\mathbf{R}) = \sum_{i=1}^{n_1} \operatorname{rank}(\mathbf{R}_i) \quad \text{while:} \quad \operatorname{rank}(\mathbf{R}_i) \leq \min(m_2, n_2), \tag{12}$$

In fact, Appendix C describes the practical implementation of SaMA adapters. We usually use small values of (n_1, m_2): $n_1 \leq \sqrt{N}, m_2 \leq \sqrt{M}$, and $n_1 m_2 \leq \min(M, N)$, where N, M denote the numbers of input and output features, respectively. Hence, in the field of deep learning applications, it is a common observation that:

$$\min(m_2, n_2) = m_2 \quad \text{and} \quad \min(m_1, n_1) = n_1,$$

Therefore, back to Eq. (12):

$$\operatorname{rank}(\mathbf{R}) = \sum_{i=1}^{n_1} \operatorname{rank}(\mathbf{R}_i) \leq \sum_{i=1}^{n_1} m_2 = n_1 m_2,$$

Similar analysis for the left factor $\mathbf{L}$, note any perfect shuffle permutation matrix $\boldsymbol{P}$ is orthogonal:

$$\Rightarrow \begin{cases} \operatorname{rank}(\boldsymbol{P}_r\mathbf{R}) = \operatorname{rank}(\mathbf{R}) \leq n_1 m_2 \\ \operatorname{rank}(\boldsymbol{P}_l^\top\mathbf{L}) = \operatorname{rank}(\mathbf{L}) \leq m_2 n_1, \end{cases} \tag{13}$$

By means of rank property for matrix multiplication, it follows that:

$$\operatorname{rank}(\Delta\mathbf{W}) = \operatorname{rank}(s\boldsymbol{P}_l^\top\mathbf{L}\boldsymbol{P}_r\mathbf{R}) \leq \min\left(\operatorname{rank}(\boldsymbol{P}_r\mathbf{R}), \operatorname{rank}(\boldsymbol{P}_l^\top\mathbf{L})\right) \leq n_1 m_2 \tag{14}$$

Generic block entries Due to this assumption, it holds almost surely (with probability 1) that the equation of rank matrix holds: $\operatorname{rank}(\mathbf{R}_i) = \min(m_2, n_2) = m_2$ as in Lemma D.1. Hence, the overall rank could be roughly expressed as: $\operatorname{rank}(\mathbf{R}) = n_1 m_2$, and similarly $\operatorname{rank}(\mathbf{L}) = m_2 n_1$. That, along with permutation matrix $\boldsymbol{P}$, leads to the empirical ranks in Eq. (13):

$$\Rightarrow \begin{cases} \operatorname{rank}(\boldsymbol{P}_r\mathbf{R}) = \operatorname{rank}(\mathbf{R}) = n_1 m_2 \\ \operatorname{rank}(\boldsymbol{P}_l^\top\mathbf{L}) = \operatorname{rank}(\mathbf{L}) = m_2 n_1, \end{cases} \tag{15}$$

Applying Sylvester's rank inequality [87, 88] then gives:

$$\begin{aligned} \operatorname{rank}(\Delta\mathbf{W}) &= \operatorname{rank}((\boldsymbol{P}_l^\top\mathbf{L})(\boldsymbol{P}_r\mathbf{R})) \\ &\geq \operatorname{rank}(\boldsymbol{P}_l^\top\mathbf{L}) + \operatorname{rank}(\boldsymbol{P}_r\mathbf{R}) - q \\ &= q + q - q = q, \qquad \text{where:} \quad q = n_1 m_2. \end{aligned} \tag{16}$$

These two Eqs. (14) and (16) imply that the generic rank of SaMA adapter is almost surely equal to $n_1 m_2$ in practice: $\operatorname{rank}(\Delta\mathbf{W}) = n_1 m_2$. □

Discussion and visualization We also illustrate the property of flexible hidden rank in Fig. 10 by means of Action space alignment scores (ASA). ASA serves as a rigorous metric that evaluates the structural expressivity of the transformation. Unlike point-wise metrics such as Cosine similarity, which are sensitive to local block-rank constraints, ASA measures the global geometric overlap between the Morpho reachable range and the target FFT subspace. It provides a pure assessment of whether the representation's total degrees of freedom are sufficient to span the required transformation space. In practice, ASA scores are given by the overlap between the subspaces spanned by their respective outputs given the same input distribution from the viewpoint of singular vectors.

A drop in the ASA scores below 1.0 indicates an absolute subspace collapse, signifying that the intrinsic rank of the transformation has surpassed the global capacity of the Morpho parameterization. At this threshold, the model encounters a hard structural bottleneck: it can no longer satisfy the stationary conditions of the FFT in the experiment of Appendix D.5, forcing a lossy projection where critical dimensions become mathematically unrecoverable regardless of the optimization effort. Meanwhile, the threshold rather supports the validation of our rank argument in this section of Appendix D.1.

D.2 Rank stability condition (Proof of Theorem 3.3)

In order to accurately define stability for an adapter with respect to rank configurations, following [33], we define the term "rank-stabilized adapter" as follows:

Definition D.1. *A SaMA adapter: $\Delta\mathbf{W} = s\boldsymbol{P}^\top\mathbf{L}\boldsymbol{P}\mathbf{R}$ in Eq. (4) is **rank-stabilized**[3] if the following conditions hold:*

1. *If the inputs to the adapter are i.i.d. such that the m'-th moment is $\boldsymbol{\Theta}_{n_1,m_2}(1)$ for each entry, then the m'-th moment of the adapter outputs is also $\boldsymbol{\Theta}_{n_1,m_2}(1)$ for each entry.*
2. *If the gradient of the loss with respect to the adapter outputs is $\boldsymbol{\Theta}_{n_1,m_2}(1)$ for each entry, then the loss gradients with respect to the adapter inputs are also $\boldsymbol{\Theta}_{n_1,m_2}(1)$ for each entry.*

In definition D.1, there are two complementary conditions together supporting fine-tuning applications. First, **Forward-pass stability (Condition 1)** ensures that the scale of hidden representations does not

[3]Because the primary configuration (n_1, m_2) directly pertains to the adapter's hidden rank, we inherit this terminology from the original work for clarity and consistency.

explode or vanish under different rank configurations during inference. If the output moments depend on (n_1, m_2), the activation distribution can shift substantially, requiring recalibration of subsequent layers or learning rates.

Second, **Backward-pass stability (Condition 2)** is crucial for gradient flow and helps avoid gradient collapse during training. It guarantees that the gradient signal passing through the adapter remains at a consistent magnitude $\mathbf{\Theta}_{n_1,m_2}(1)$. Without this property, instability in gradient flow makes it difficult to find a universal hyperparameter setting across architectural scales. Both of the two conditions play significant roles that preserve signal energy magnitudes and stabilize computational graphs throughout all layers during training and inference time.

Unlike rsLoRA [33], which uses analysis in the limiting regime when $r \to \infty$, we only examine SaMA adapters in normal settings and prove that the scaling of s (sometimes being denoted as s_{SaMA} for clarity) that could achieve *rank-stabilized* adapters is:

$$s_{\text{SaMA}} = n_1^{0.25}, \tag{17}$$

for some scalar $\alpha \in \mathbb{R}$.

Theorem (Detailed statement of Theorem 3.3)**.** *Consider SaMA adapters which are of the form $\Delta\mathbf{W} = s\boldsymbol{P}_l^\top \mathbf{L}\boldsymbol{P}_r\mathbf{R}$, where $\boldsymbol{P}_{l,r}$ is the non-trainable perfect shuffle permutation matrix, two trainable factors $\mathbf{L}$ of shape $m_2m_1 \times m_2n_1$, $\mathbf{R}$ of shape $n_1m_2 \times n_1n_2$ are block-diagonal matrices as defined in Eq. (5) – where (M, N) are large and do not depend on the configuration (n_1, m_2), initialized such that $\mathbf{L}$ is a zero matrix $\mathbf{0}_{m_2m_1\times m_2n_1}$, entries of $\mathbf{R}$ are i.i.d with mean $\mathbf{0}$ and variance $\sigma^2_{\mathbf{R}}$ and scaling factor s (or s_{n_1,m_2}) $\in \mathbb{R}$; and train the adapter for finite n SGD updates.*

In expectation over the initialization method, all adapters are rank-stabilized if and only if:

$$s_{\text{SaMA}} = \Theta\left(n_1^{1/4}\right), \tag{18}$$

The above theorem is a detailed description and explanation of Theorem 3.3. It suggests that there exists a feasible way to ensure stability through the adapters regardless of arbitrary rank configurations is to adapt the scaling factor: $s_{\text{SaMA}} \propto n_1^{0.25}$.

The role of scaling and multi-layer stability A critical insight from Theorem 3.3 is that maintaining stability across arbitrary "rank configuration" of (n_1, m_2) necessitates scaling the adapter by: $s_{\text{SaMA}} \propto n_1^{0.25}$. While it may initially seem that activations and gradients in middle layers would inevitably depend on (n_1, m_2) due to the influence of preceding (forward pass) or succeeding (backward pass) adapters, this dependency is mitigated through an inductive flow:

- **Forward propagation** Since the intial input data is independent of (n_1, m_2), applying the rank-stabilization theorem layer-by-layer from input to output guarantees that the activation magnitude remains stable ($\mathbf{\Theta}_{n_1,m_2}(1)$) throughout the entire forward pass.
- **Backward propagation** Similarly, by starting from the loss gradient at the output and moving backward, the theorem ensures that gradient magnitudes do not explode or vanish, preserving a consistent scale of $\mathbf{\Theta}_{n_1,m_2}(1)$ across all adapters.

Limitation note It is also important to distinguish between numerical stability and feature quality. The theorem only ensures that the learning process does not fall into a collapse mode in the limiting regime of "rank configuration" (n_1, m_2), but it does not concentrate and establish a connection between heavy-mode configuration and better features. To this end – the quality of trained features and the practical benefits of the corrected scaling factor s_{SaMA} may require further analysis and validation.

Lemma D.2. *Given a set $\mathbf{X} = \{\mathbf{X}_K\}$ that contains K independent and identically distributed (i.i.d) random variables with mean $\mathbb{E}[\mathbf{X}_k] = 0, \text{Var}(\mathbf{X}_k) = \sigma_X^2 > 0, \forall k$, and high-order absolute moments $\mathbb{E}[|\mathbf{X}_k|^m], \forall m \geq 2$ are finite.*

Let $\mathbf{S} = \sum_{m=1}^{K} \mathbf{X}_k$ be the sum of these random variables. As K grows large, the m-th absolute moment of $\mathbf{S}$ follows the asymptotic scaling:

$$\mathbb{E}\left[|\mathbf{S}|^m\right] = \Theta\left(K^{m/2}\text{Var}(\mathbf{X}_k)^{m/2}\right) = \Theta\left(K^{m/2}\sigma_X^m\right). \tag{19}$$

Proof of Lemma D.2. By Rosenthal inequality [89] for independent zero-mean random variables, there exists a constant C_m such that the upper bound:

$$\mathbb{E}\left[|\mathbf{S}|^m\right] \leq C_m \max\left\{\sum_{k=1}^{K}\mathbb{E}\left[|\mathbf{X}_k|^m\right], \left(\sum_{k=0}^{K}\mathbb{E}\left[\mathbf{X}_k^2\right]\right)^{m/2}\right\}, \tag{20}$$

Since the variances are i.i.d, we analyze the scaling rate with respect to K for both components inside the $\max$ function.

- First component: $\sum_{k=1}^{K}\mathbb{E}\left[|\mathbf{X}_k|^m\right] = K\,\mathbb{E}\left[|\mathbf{X}_1|^m\right] = \Theta(K)$.
- Second component: $\left(\sum_{k=0}^{K}\mathbb{E}\left[\mathbf{X}_k^2\right]\right)^{m/2} = (K\sigma_X^2)^{m/2} = K^{m/2}\sigma^m$.

  As $K \to \infty$, given $p \geq 2$, the $K^{m/2}\sigma^m$ term strictly dominates the linear term $\Theta(K)$, apply back to Eq. (20), we establish the exact asymptotic bound as stated. □

Proof of Theorem 3.3. **Proof sketch** With a constant initialization variance σ_R, we estimate the variance in $\mathbf{L}$ update. Deriving the scaling factor s is fundamentally an algebraic balancing act to neutralize this structural discrepancy, bounding both flows at $\Theta(1)$. Finally, under independence assumptions and Lemma D.2, successfully stabilizing the 2nd-order moment (variance) serves as a sufficient condition to generalize and bound high-order absolute moments.

R-branch Forward pass Consider the intermediate activation vector $\boldsymbol{z} = \mathbf{R}\boldsymbol{x}$. Each vector feature $\boldsymbol{z}_k$ is a sum of n_2 independent elements.

For simplicity, we first take the 2nd moment into account then make a generalization of high-order moments. Apply Lemma D.2, given $\Theta_{n_1,m_2}(\sigma_R) = \Theta(1)$:

$$\mathbb{E}\left[|\boldsymbol{z}_k|^m\right] = \Theta\left(n_2^{p/2}(\sigma_R)^m\right) \tag{21}$$

$$\Rightarrow \mathbb{E}[\boldsymbol{z}_k^2] = \Theta(n_2) = \Theta(n_1^{-1}) = \mathbb{E}[\boldsymbol{z}^2]. \tag{22}$$

L-block update: Due to the sparsity constraint, only intra-block boundaries accumulate gradients, at time step n. $\mathbf{L}_{i,k}^{(n)} \propto \sum_{t=1}^{n} s \cdot \nabla \boldsymbol{y}_i^{(t)} \cdot \boldsymbol{z}_k^{(t)}$. With finite n steps and $\Theta(1)$ error gradients, zero-mean distribution, the variance of a *valid element* in $\mathbf{L}$ is:

$$\begin{aligned}\sigma_L^{2(n)} = \mathbb{E}[|\mathbf{L}_{i,k}^{(n)}|^2] &= \mathbb{E}\left[|n \cdot s \cdot \nabla \boldsymbol{y}_i \cdot \boldsymbol{z}_k|^2\right] \\ &= \Theta\left(n^2 \cdot s^2 \cdot \mathbb{E}[|\nabla \boldsymbol{y}_i|^2] \cdot \mathbb{E}[|\boldsymbol{z}|^2]\right), \\ \Rightarrow \sigma_L^2 &= \Theta(s^2 \cdot \sigma_z^2) = \Theta(s^2 n_1^{-1})\end{aligned} \tag{23}$$

Output condition for $\boldsymbol{y}$: With the assumption that $\boldsymbol{y}, \boldsymbol{z}_k$ are independent, combined with zero mean distributions: $\mathrm{Var}(\mathbf{L}_{i,k}\boldsymbol{z}_k) = \mathrm{Var}(\mathbf{L}_{i,k})\mathrm{Var}(\boldsymbol{z}_k)$.

Each generic formula of output feature $\boldsymbol{y}_i = s\sum_{k=1}^{n_1}\mathbf{L}_{i,k}\boldsymbol{z}_k$[4] is a sum of n_1 components. Due to accumulated updates, assume that $\mathbf{L}, \boldsymbol{z}$ are independent of each other. The output variance is:

$$\begin{aligned}\mathbb{E}[|\boldsymbol{y}_i|^2] &= s^2 \cdot \mathbb{E}\left[\mathbf{L}_{i,k}\boldsymbol{z}_k\right] \\ &= \Theta\left(s^2 \cdot n_1 \mathrm{Var}(\mathbf{L}_{i,k}\boldsymbol{z}_k)\right) \\ \Rightarrow \sigma_y^2 &= \Theta\left(s^2 n_1 \cdot \mathrm{Var}(\mathbf{L}_{i,k})\mathrm{Var}(\boldsymbol{z}_k)\right)\end{aligned} \tag{24}$$

Since: $\sigma_L^2 = \Theta(s^2 \cdot \sigma_z^2) = \Theta(s^2 n_1^{-1})$, $\mathrm{Var}(\boldsymbol{z}) = \mathbb{E}[\boldsymbol{z}^2] - 0 = \mathbb{E}[\boldsymbol{z}^2] = \Theta(n_1^{-1})$:

$$\sigma_y^2 = \mathbb{E}[|\boldsymbol{y}_i|^2] = \Theta(s^2 n_1 \cdot s^2 n_1^{-1} \cdot n_1^{-1}) = \Theta(s^4 \cdot n_1^{-1}) \tag{25}$$

To satisfy the stabilization constrain $\mathbb{E}[\boldsymbol{y}_i^2] = 1$, we strictly require $s^4 \cdot n_1^{-1} = \textit{const}$, leading to the amplification factor: $s \propto n_1^{0.25}$.

Backward gradient condition for $\nabla \boldsymbol{x}$: With the assumption that $\mathbf{R}, \boldsymbol{v}_k$ are independent, combined with zero mean distributions: $\mathrm{Var}(\mathbf{R}_{i,k}\boldsymbol{v}_k) = \mathrm{Var}(\mathbf{R}_{i,k})\mathrm{Var}(\boldsymbol{v}_k)$.

[4] It is indeed a span of n_1 elements, we denote from 1 to n_1 for simplicity.

Let $\boldsymbol{v}_k = \sum_{i=1}^{m_1} \mathbf{L}_{k,i}^\top \nabla \boldsymbol{y}_i$. The input gradient is:

$$\nabla \boldsymbol{x}_j = s \sum_{k=1}^{m_2} \mathbf{R}_{j,k}^\top \boldsymbol{v}_k, \tag{26}$$

With $\Theta(\nabla_{\boldsymbol{y}_i}) = 1$, the 2nd-moment, equivalent to the variance of $\boldsymbol{v}_k$ is:

$$\sigma_v^2 = \mathbb{E}[\boldsymbol{v}_k^2] = \Theta(m_1 \cdot \sigma_{\mathbf{L}}^2 \cdot 1) = \Theta(m_1 n_1^{-1} s^2) \tag{27}$$

The 2nd-moment of input gradient, summed over m_2 terms:

$$\begin{aligned} \sigma_{\nabla \boldsymbol{x}}^2 &= \mathbb{E}[\nabla \boldsymbol{x}_i^2] = \Theta(s^2 \cdot m_2 \cdot \sigma_R^2 \cdot \sigma_v^2) \\ &= \Theta(s^2 \cdot m_2 \cdot 1 \cdot m_1 n_1^{-1} s^2) = \Theta(s^4 n_1^{-1}), \text{ since: } m_1 m_2 = M. \end{aligned} \tag{28}$$

Similarly, to satisfy the stabilization constraint, the amplification factor: $s \propto n_1^{0.25}$.

Generalization to $m-th$ moments: Because the internal tensors are formed by summing multiple independent mean-zero random variables, Lemma 1 establishes that the asymptotic scaling of all m-th absolute moments ($m \geq 2$) is strictly governed by a power of the variance: $\mathbb{E}[|S|^m] = \Theta(K^{m/2}\sigma^m)$. Since we have proved that the variance (2nd moment) of all evaluated tensors ($\boldsymbol{y}$ and $\nabla \boldsymbol{x}$) is strictly stabilized at $\Theta(1)$, all higher-order m-th absolute moments are inherently bounded at $\Theta(1)$ as well. □

D.3 Expressivity in low effective rank. Proof of Theorem 3.2

We restate the two competing approximations precisely. Let $\mathcal{F}^{(r)}$ denote the manifold of Morpho matrices with configuration $(n_1, m_2) = (r, r)$, and define:

$$\mathbf{W}_L \coloneqq \underset{\substack{U = \mathbf{BA} \\ \mathbf{B}, \mathbf{A}^\top \in \mathbb{R}^{d \times r}}}{\arg\min} \|\mathbf{W} - U\|_F, \qquad \mathbf{W}_M \coloneqq \underset{U \in \mathcal{F}^{(r)}}{\arg\min} \|\mathbf{W} - U\|_F. \tag{29}$$

$\mathbf{W}_L$ is the Eckart–Young rank-r truncation; $\mathbf{W}_M$ is the optimal Morpho approximation at the same parameter budget. The proof proceeds in four steps: (i) show that $\mathcal{F}^{(r)}$ contains a canonical family of block-structured matrices (Lemma D.3); (ii) establish that the residual of any block-rank-1 approximation is orthogonal to the approximation itself (Lemma D.4); (iii) derive a rank-dependent lower bound on the energy captured by $\mathbf{W}_M$; (iv) identify the spectral crossover threshold via the Marchenko–Pastur law[90] (Lemma D.6 and Corollary D.1).

Step 1: Morpho Contains All Block-Rank-1 Matrices

Lemma D.3 (Block-rank-1 inclusion)**.** *Let $\mathbf{W} \in \mathbb{R}^{d \times d}$ be partitioned into a uniform $r \times r$ grid of blocks $\mathbf{W}_{jk} \in \mathbb{R}^{s \times s}$, $s = d/r$. Denote by $\mathbf{M}_{jk} = \boldsymbol{u}_{jk}\boldsymbol{v}_{jk}^\top$ the optimal rank-1 approximation of each block. Then the block-assembled matrix $\mathbf{M}^{(s)} = \sum_{j,k} \mathbf{M}_{jk}$ lies exactly in $\mathcal{F}^{(r)}$:*

$$\mathbf{M}^{(s)} \in \mathcal{F}^{(r)}, \qquad (n_1, m_2) = (r, r). \tag{30}$$

Proof. We construct $(\mathbf{L}, \mathbf{R})$ explicitly. Let $\boldsymbol{u}'_{jk} = \sigma_{jk}^{1/2}\boldsymbol{u}_{jk}$ and $\boldsymbol{v}'_{jk} = \sigma_{jk}^{1/2}\boldsymbol{v}_{jk}$, where $\sigma_{jk} = \sigma_1(\mathbf{W}_{jk})$.

Factor assignment. Assign the j-th block of $\mathbf{L} \in \mathcal{BD}$ (of size $s \times r$) and the k-th block of $\mathbf{R} \in \mathcal{BD}$ (of size $r \times s$) as:

$$\mathbf{L}_{:,k}^{(j)} \leftarrow \boldsymbol{u}'_{j,k}, \qquad \mathbf{R}_{j,:}^{(k)} \leftarrow (\boldsymbol{v}'_{j,k})^\top, \qquad \forall\, 1 \leq j, k \leq r. \tag{31}$$

Each block of $\mathbf{L}$ pools the r left singular vectors associated with its grid row; each block of $\mathbf{R}$ pools the r right singular vectors associated with its grid column.

Inner routing via $\boldsymbol{P}_r$. The permutation $\boldsymbol{P}_r$ acts as a bipartite switchboard: it pairs precisely the k-th column of $\mathbf{L}^{(j)}$ with the j-th row of $\mathbf{R}^{(k)}$, synthesizing the outer product $\boldsymbol{u}'_{jk}(\boldsymbol{v}'_{jk})^\top = \mathbf{M}_{jk}$. Over all r^2 pairs, $\mathbf{L}\boldsymbol{P}_r\mathbf{R}$ simultaneously generates all r^2 rank-1 blocks.

Spatial routing via P_l. The outer permutation P_l is a pure reindexing: it relocates the r^2 rank-1 blocks to their correct (j, k) positions in the $r \times r$ grid, recovering $\mathbf{M}^{(s)}$ exactly.

Since $\mathbf{L}$ and $\mathbf{R}$ are unconstrained beyond their block-diagonal structure, and the permutations are fixed linear bijections, any choice of $\{\boldsymbol{u}'_{jk}, \boldsymbol{v}'_{jk}\}$ is realizable, establishing $\mathbf{M}^{(s)} \in \mathcal{F}^{(r)}$. □

Step 2 : Orthogonality of Block-Rank-1 Residuals

Lemma D.4 (Block-wise Pythagorean identity)**.** *With $\mathbf{M}^{(s)}$ as in Lemma D.3, the approximation residual $\mathbf{W} - \mathbf{M}^{(s)}$ is Frobenius-orthogonal to $\mathbf{M}^{(s)}$:*

$$\langle \mathbf{W} - \mathbf{M}^{(s)}, \mathbf{M}^{(s)} \rangle_F = 0 \quad \Longrightarrow \quad \|\mathbf{W} - \mathbf{M}^{(s)}\|_F^2 = \|\mathbf{W}\|_F^2 - \|\mathbf{M}^{(s)}\|_F^2. \tag{32}$$

Proof. The inner product decomposes blockwise: $\langle \mathbf{W} - \mathbf{M}^{(s)}, \mathbf{M}^{(s)} \rangle_F = \sum_{j,k} \langle \mathbf{W}_{jk} - \mathbf{M}_{jk}, \mathbf{M}_{jk} \rangle_F$. Each term vanishes because $\mathbf{M}_{jk} = \sigma_{jk} \boldsymbol{u}_{jk} \boldsymbol{v}_{jk}^\top$ is the optimal rank-1 approximation of $\mathbf{W}_{jk}$: by Eckart–Young, the residual $\mathbf{W}_{jk} - \mathbf{M}_{jk}$ lies in the orthogonal complement of $\mathrm{span}(\boldsymbol{u}_{jk}) \otimes \mathrm{span}(\boldsymbol{v}_{jk})$, which contains $\mathbf{M}_{jk}$. Summing over all blocks yields Eq. (32). □

Step 3 : Energy Comparison and the Crossover Threshold Existence

Ideas We show that each strip of r rows in $\mathbf{W}$ contains at least an energy unit under this specific condition $R > d - s (s = d/r)$, and Morpho matrices are able to capture the energy.

Proof of Theorem 3.2. Define the *energy* of a matrix as its squared Frobenius norm: $\mathbf{E}(\mathbf{W}) = \|\mathbf{W}\|_F^2 = \sum_i \sigma_i^2(\mathbf{W})$. Minimizing approximation error is equivalent to maximizing captured energy: $\|\mathbf{W} - \mathbf{W}_Q\|_F^2 = \mathbf{E}(\mathbf{W}) - \mathbf{E}(\mathbf{W}_Q)$.

Energy ceiling for low-rank. By the Eckart–Young–Mirsky theorem, the rank-r truncation captures exactly the r largest singular values. Since $\mathbf{W}$ has rank R with all unit singular values, $\sigma_i(\mathbf{W}) = \mathbf{1}[i \le R]$, so:

$$\mathbf{E}(\mathbf{W}_L) = \sum_{i=1}^{r} \sigma_i^2(\mathbf{W}) = r. \tag{33}$$

This is a hard ceiling: no rank-r matrix can capture more than r units of energy, regardless of the target structure.

Energy floor for Morpho. Partition $\mathbf{W}$ into its $r \times r$ grid of $s \times s$ blocks ($s = d/r$), and let $\mathbf{M}^{(s)}$ be the block-rank-1 approximation from Lemma D.3. Since $\mathbf{W}_M$ is the global optimum over $\mathcal{F}^{(r)} \ni \mathbf{M}^{(s)}$:

$$\mathbf{E}(\mathbf{W}_M) \ge \mathbf{E}(\mathbf{M}^{(s)}) = \sum_{j=1}^{r} \sum_{k=1}^{r} \sigma_1^2(\mathbf{W}_{jk}). \tag{34}$$

Consider $\mathbf{W}$ as r row strips (by merging horizontally r blocks of s-by-s), where each strip $\mathbf{W}_{\text{strip}}^{(j)}$ consists of s rows. Let $\mathbf{W}_{\text{rest}}^{(j)}$ denote the remaining $d - s$ rows of $\mathbf{W}$. We analyze the energy captured by Morpho within a single strip.

Define two vector spaces in $\mathbb{R}^d$:

- $\mathrm{A} = \mathrm{Row}\,(\mathbf{W})$. Since $\mathrm{rank}(\mathbf{W}) = R$, we have $\dim(A) = R$.
- $\mathrm{B} = \mathrm{Null}(\mathbf{W}_{\text{rest}}^{(j)})$. By the Rank-Nullity theorem:

$$\dim(\mathrm{B}) = d - \mathrm{rank}(\mathbf{W}_{\text{rest}}^{(j)}) \ge d - (d - s) = s. \tag{35}$$

By Grassmann's formula, the intersection of these subspaces satisfies:

$$\dim(\mathrm{A} \cap \mathrm{B}) \ge \dim(\mathrm{A}) + \dim(\mathrm{B}) - \dim(\mathrm{A} \cup \mathrm{B}) \ge R + s - d. \tag{36}$$

If we impose the condition $R > d - s$, then $\dim(\mathrm{A} \cap \mathrm{B}) \ge 1$. Thus, there exists a non-zero vector $\boldsymbol{x} \in \mathrm{A} \cap \mathrm{B}$. Assume $\boldsymbol{x}$ is normalized such that $\|\boldsymbol{x}\| = 1$.

Since $\boldsymbol{x} \in \mathrm{A} = \mathrm{Row}(\mathbf{W})$, it is strictly orthogonal to $\mathrm{Null}(\mathbf{W})$. Because all non-zero singular values of $\mathbf{W}$ are equal to 1, $\mathbf{W}$ acts as an isometry on its row space, preserving the vector energy completely: $\|\mathbf{W}\boldsymbol{x}\| = \|x\| = 1$.

Simultaneously, since $\boldsymbol{x} \in \mathrm{B} = \mathrm{Null}(\mathbf{W}^{(j)}_{\text{rest}})$, we have $\mathbf{W}^{(j)}_{\text{rest}}\boldsymbol{x} = 0$.

The squared norm of the full transformation $\mathbf{W}\boldsymbol{x}$ decomposes exactly over the strip and the rest of the matrix:

$$\|\mathbf{W}^{(j)}_{\text{strip}}\boldsymbol{x}\|^2 + \|\mathbf{W}^{(j)}_{\text{rest}}\boldsymbol{x}\|^2 = \|\mathbf{W}\boldsymbol{x}\|^2 \Rightarrow \|\mathbf{W}^{(j)}_{\text{strip}}\boldsymbol{x}\|^2 + 0 = 1. \tag{37}$$

By the definition of the spectral norm, this implies $\sigma_1^2(\mathbf{W}^{(j)}_{\text{strip}}) \geq 1$.

Regarding Morpho parameterization. Now partition the strip $\mathbf{W}^{(j)}_{\text{strip}}$ into r constituent blocks of $(s \times s)$: $\mathbf{W}_{j1}, \mathbf{W}_{j2}, \ldots, \mathbf{W}_{jr}$.

Lemma D.5 (Block spectral norm)**.** *Let $\mathbf{W}_j$ be a matrix and its partitioned horizontally into r sub-matrices: $\mathbf{W}_j = [\mathbf{W}_{j1}, \mathbf{W}_{j2}, \ldots, \mathbf{W}_{jr}]$ (as being pre-defined in the section above).*

The squared spectral norm of the concatenated matrix $\mathbf{W}_j$ is bounded the by sum of the squared spectral norms of its individual blocks:

$$\sigma_1^2\left(\mathbf{W}_j^{(j)}\right) \leq \sum_{k=1}^{r} \sigma_1^2(\mathbf{W}_{jk}). \tag{38}$$

Proof sketch: We show this by using spectral norm properties and Cauchy-Schwarz inequality.

Proof of Lemma D.5. By the definition of the spectral norm, there exists a unit vector $\boldsymbol{y}$ with $\|\boldsymbol{y}\| = 1$ such that $\sigma_1^2(\mathbf{W}_j) = \|\mathbf{W}_j\boldsymbol{y}\|^2$. We partition $\boldsymbol{y}$ into r segments $\boldsymbol{y} = [\boldsymbol{y}_1^\top, \boldsymbol{y}_2^\top, \ldots, \boldsymbol{y}_r^\top]^\top$ conforming to the structure of $\mathbf{W}_j$, which implies $\sum_{k=1}^{r} \|\boldsymbol{y}_k\|^2 = 1$.

Decomposing the linear transformation and applying the triangle inequality, we obtain:

$$\|\mathbf{W}_j\boldsymbol{y}\| = \|\sum_{k=1}^{r} \mathbf{W}_{jk}\boldsymbol{y}_k\| \leq \sum_{k=1}^{r} \|\mathbf{W}_{jk}\boldsymbol{y}_k\| \leq \sum_{k=1}^{r} \sigma_1(\mathbf{W}_{jk})\|\boldsymbol{y}_k\|. \tag{39}$$

Squaring both sides and applying the Cauchy-Schwarz inequality yields:

$$\begin{aligned} \|\mathbf{W}_j\boldsymbol{y}\|^2 &\leq \left(\sum_{k=1}^{r} \sigma_1^2(W_k)\right)\left(\sum_{k=1}^{r} \|\boldsymbol{y}_k\|^2\right), \\ \Rightarrow \sigma_1^2(\mathbf{W}_j) = \sigma_1^2(\mathbf{W}^{(j)}_{\text{strip}}) &\leq \sum_{k=1}^{r} \sigma_1^2(\mathbf{W}_{jk}), \end{aligned} \tag{40}$$

Since $\sum_{k=1}^{r} \|\boldsymbol{y}_k\|^2 = 1$, this completes the proof. □

By applying Lemma D.5:

$$\sum_{k=1}^{r} \sigma_1^2(\mathbf{W}_{jk}) \geq \sigma_1^2(\mathbf{W}^{(j)}_{\text{j}}). \tag{41}$$

Since the Morpho approximation $\mathbf{W}_M$ can independently formulate an optimal rank-1 approximation for every block (established in Lemma D.3), it captures at least this much energy per strip. Summing over all r strips yields:

$$\mathbf{E}(\mathbf{W}_M) \geq \sum_{j=1}^{r}\sum_{k=1}^{r} \sigma_1^2(\mathbf{W}_{jk}) \geq r. \tag{42}$$

Crossover condition. Combining the ceiling and the floor, we conclude that $\mathbf{E}(\mathbf{W}_M) \geq \mathbf{E}(\mathbf{W}_L)$ almost surely holds whenever $R > d - s$. This establishes a deterministic worst-case threshold $R_0 \leq d - s$, confirming existence of a nontrivial finite crossover.

However, because this worst-case bound is still extremely close to d, we next employ random matrix theory to derive the average-case crossover threshold, which governs practical deep learning scenarios. □

Step 4 : Quantifying the Crossover via Random Matrix Theory

The proof above establishes existence of R_0, but for the worst-case. We now investigate practical applications on random matrix using the Marchenko–Pastur law [90] and the Bai-Yin theorem [91].

Lemma D.6 (Recall of matrix norms for random matrix)**.** *Let $\mathbf{S} \in \mathbb{R}^{s\times s}$ have i.i.d. $\mathcal{N}(0, \sigma_S^2)$ entries. The following statements hold:*

- $\|\mathbf{S}\|_F^2 = s^2\sigma_S^2$.
- $\sigma_1^2(\mathbf{S}) \xrightarrow{s\to\infty} C(s)\sigma_S^2$, *where $C(s)$ is a constant depending on size s only. When s is sufficiently large, $C(s) \to 4$.*

Proof of Lemma D.6. By the law of large numbers, $\|\mathbf{S}\|_F^2 = \sum_{ij} S_{ij}^2 \to s^2\sigma_S^2$ as $s \to \infty$.

For the spectral norm, treat $\mathbf{G} = s^{-1}\mathbf{S}\mathbf{S}^\top$ as a sample covariance matrix with aspect ratio $\lambda = s/s = 1$. The Marchenko–Pastur law [90] places eigenvalues $\lambda_i(\mathbf{G})$ in $[\sigma_S^2(1-\sqrt{\lambda})^2,\ \sigma_S^2(1+\sqrt{\lambda})^2]$; at $\lambda = 1$ the upper edge is $4\sigma_S^2$, so $\lambda_{\max}(\mathbf{G}) \to 4\sigma_S^2$, giving:

$$\sigma_1^2(\mathbf{S}) = s\,\lambda_{\max}(\mathbf{G}) \xrightarrow{s\to\infty} 4s\sigma_S^2. \tag{43}$$

From now on, we refer $C(s) \leftrightarrow C$ interchangeably for simplicity.

Remark 1 (Finite-size Corrections and Tracy-Widom Fluctuations). While the Bai-Yin theorem establishes $C(s) \to 4$ in the asymptotic limit ($s \to \infty$), practical Morpho configurations employ finite block sizes (e.g., $s \in [64, 512]$ for $d = 4096$). For finite-dimensional random matrices, the largest eigenvalue exhibits statistical fluctuations around the Marchenko-Pastur upper edge, governed by the Tracy-Widom distribution [92]. □

Step 4a: Asymptotic Gaussianity of Local Windows. While the worst-case bound $R_0 \le d - s$ assumes an adversarial energy distribution, we now concentrate on matrices exhibiting random-like properties. Let $\mathbf{W} = \mathbf{U}\mathbf{V}^\top$ where $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{d\times R}$ are drawn from the Haar measure on the Stiefel manifold. Consider a local $s \times s$ sub-block $\mathbf{W}_{jk} = \mathbf{U}_j\mathbf{V}_k^\top$, where $\mathbf{U}_j, \mathbf{V}_k \in \mathbb{R}^{s\times R}$ are the corresponding row slices, $j, k \in \overline{1..r}$. Each entry of this block is computed as an inner product of R-dimensional vectors:

$$(\mathbf{W}_{jk})_{ab} = \langle \mathbf{U}_{j,a,:}, \mathbf{V}_{k,b,:} \rangle\,. \tag{44}$$

Because the full matrices $\mathbf{U}$ and $\mathbf{V}$ have orthonormal columns distributed uniformly in $\mathbb{R}^d$, the expected squared magnitude of any single element is $\mathbb{E}[\mathbf{U}_{j,ac}^2] = \mathbb{E}[\mathbf{V}_{k,bc}^2] = 1/d$. Since $\mathbf{U}$ and V are independent, the variance of their scalar product is $\mathbb{E}[\mathbf{U}\mathbf{V}] = 1/d^2$. Summing over R independent components yields the theoretical variance for the entries of the sub-block:

$$\sigma_H^2 = \mathrm{Var}\left((\mathbf{W}_{jk})_{ab}\right) = R\left(\frac{1}{d^2}\right) = \frac{R}{d^2}. \tag{45}$$

By classical results on Haar-distributed matrices and the Central Limit Theorem, when R increases, the entries of $\mathbf{W}_{jk}$ asymptotically converge to independent and identically distributed (i.i.d.) Gaussian variables $\mathcal{N}(0, \sigma_H^2)$. The deviation from the true Gaussian measure in total variation distance diminishes rapidly when R, d are sufficiently large. We empirically show this effect on a local window in Fig. 6, confirming our evidence for random-like matrix analysis.

Step 4b: Marchenko-Pastur Law and Average-Case Crossover. Because W_{jk} behaves as a random Gaussian matrix, the spectrum of its normalized sample covariance follows the Marchenko-Pastur law with an aspect ratio $\lambda = s/s = 1$. According to the Bai-Yin theorem [91], the maximum singular value squared of the block, $\sigma_1^2(W_{jk})$, converges almost surely to the upper edge of the Marchenko-Pastur support, with respect to Lemma D.6:

$$\sigma_1^2(\mathbf{W}_{jk}) \approx Cs\sigma_H^2 = Cs\sigma_H^2 = Cs\frac{R}{d^2}. \tag{46}$$

where constant $C \xrightarrow{s\to\infty} (1+\sqrt{\lambda})^2 = 4$.

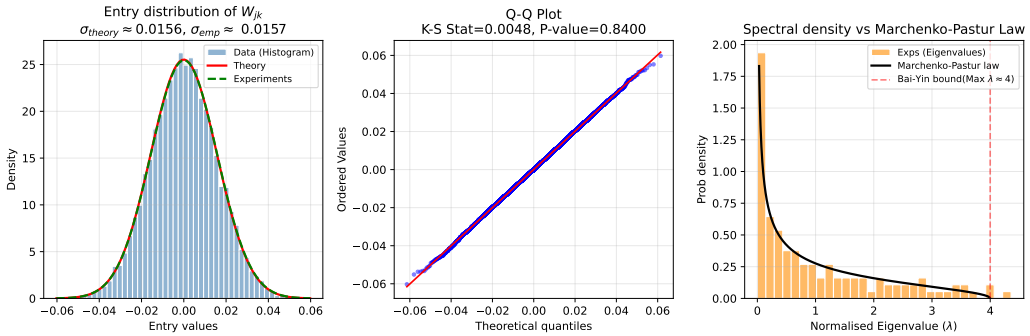

Figure 6: Statistical and spectral validation of local sub-blocks $\mathbf{W}_{jk}$ extracted from a Haar-distributed target matrix. **(Left)** Empirical probability density of the entries from an $s \times s$ sub-block $\mathbf{W}_{jk}$. The empirical distribution closely aligns with the theoretical Gaussian density $\mathcal{N}(0, R/d^2)$. **(Middle)** The Q-Q plot confirms the asymptotic normality of the entries, corroborated by a high Kolmogorov-Smirnov test p-value. **(Right)** The empirical spectral density of the normalized sample covariance matrix perfectly follows the theoretical Marchenko-Pastur law. The maximum eigenvalue strictly adheres to the Bai-Yin upper bound at $\lambda \approx 4$, establishing the rigorous mathematical foundation for Morpho's average-case energy capture threshold at $R_0 \approx d/4$.

Recall from Lemma D.3 that the Morpho parameterization can independently formulate an optimal rank-1 approximation for every block. Consequently, the total expected energy captured by Morpho across all r^2 blocks is:

$$\mathbf{E}(\mathbf{W}_M) \geq \sum_{j=1}^{r}\sum_{k=1}^{r} \sigma_1^2(\mathbf{W}_{jk}) \approx r^2 \left(Cs\frac{R}{d^2} \right) = Cr\frac{R}{d}. \tag{47}$$

For Morpho to strictly outperform the LoRA baseline, its captured energy must exceed LoRA's deterministic energy ceiling ($\mathbf{E}(\mathbf{W}_L) = r$). Solving $\mathbf{E}(\mathbf{W}_M) > \mathbf{E}(\mathbf{W}_L)$ yields:

$$Cr\frac{R}{d} > r \iff R > \frac{d}{C} \tag{48}$$

This average-case threshold demonstrates that under standard random initialization or well-regularized weight distributions, Morpho efficiently surpasses LoRA at approximately 25% of the full rank. This theoretical derivation tightly mirrors our empirical observations of the crossover point in practical settings in Figs. 2 and 14.

Corollary D.1 (Universal crossover bound)**.** *With respect to the above settings, the crossover threshold R_0 satisfies:*

$$R_0 = \lceil \frac{d}{C} \rceil, \tag{49}$$

and is independent of the budget rank r. *Consequently, for any fixed matrix dimension d, the fraction of targets for which Morpho dominates low-rank approximation is at least $1 - 1/C$, converging to 0.75, as $d \to \infty$.*

Proof. This is a direct corollary from Eq. (48) and Lemma D.6 by means of the Marchenko–Pastur law [90] and the Bai-Yin theorem [91].

There is no dependence on budget r, making R_0 a function of d and C alone. As a result of Lemma D.6, the actual threshold might vary since C follows Tracy-Widom distribution [92]. □

Synthesis. Corollary D.1 reveals a structurally important universality: the rank at which Morpho overtakes low-rank approximation is determined entirely by the ambient dimension d and the spectral concentration constant C—not by the parameter budget r. This means that as models scale (larger d), the crossover threshold grows proportionally, but the *fraction* of high-eRank targets for which Morpho dominates remains constant. In the regime of LLMs, where weight matrices are large and their updates are empirically observed to have broadly distributed singular spectra [93], this universality directly predicts the practical advantage of Morpho observed in Sec. 3.2.

D.4 Numerical stability of SaMA over ranks

Theorem 3.3 establishes the optimal scaling factor for SaMA as:

$$s_{\text{SaMA}} = \Theta\left(n_1^{1/4}\right), \tag{50}$$

which ensures numerical stability of output and gradient magnitudes as the hidden dimension grows. We now empirically validate this prescription against a grid of different scaling strategies.

Experimental setup. We fix matrix dimensions $M = N = 2048$ and a constant learning rate, then independently vary n_1 (with m_2 fixed) and m_2 (with n_1 fixed), monitoring output norms and gradient magnitudes under six scaling schemes parameterized by the free variable $v \in \{n_1, m_2\}$:

$$s_{\text{SaMA}} \in \left\{\Theta(1),\ \Theta\left(v^{-1}\right),\ \Theta\left(v^{-1/2}\right),\ \Theta\left(v^{-1/4}\right),\ \Theta\left(v^{1/4}\right),\ \Theta\left(v^{1/2}\right)\right\}. \tag{51}$$

Results. Figure 7 reports the outcome across both variation axes, with y-axes representing variance of output $\boldsymbol{y}$ (forward pass) and gradient $\nabla_{\mathcal{L}}\boldsymbol{x}$ (backward pass), respectively. Two scaling schemes maintain stable signal propagation; all others exhibit either systematic attenuation or divergence as the free parameter grows:

- **Panels (a,b) — varying m_2, fixed n_1:** the constant scheme $s = \Theta(1)$ (green) is the unique stable choice, consistent with the theoretical prediction that s has no dependence on m_2.
- **Panels (c,d) — varying n_1, fixed m_2:** the quarter-power scheme $s = \Theta(n_1^{1/4})$ (purple) is the unique stable choice, directly confirming Theorem 3.3.

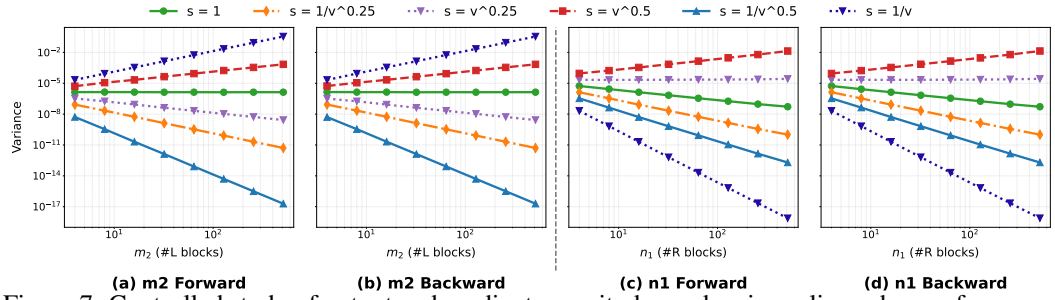

Figure 7: Controlled study of output and gradient magnitudes under six scaling schemes for s_{SaMA}, across two hyperparameter axes. **(a,b):** n_1 fixed, m_2 varying — $s = \Theta(1)$ **(green)** is the unique stable scheme. **(c,d):** m_2 fixed, n_1 varying — $s = \Theta(n_1^{0.25})$ **(purple)** is the unique stable scheme. All other schemes exhibit monotone attenuation or divergence.

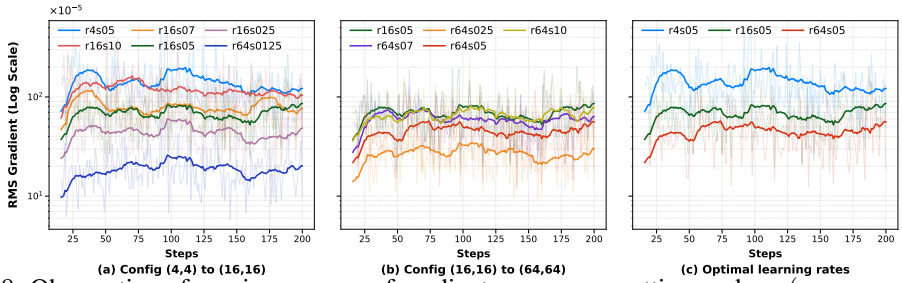

Figure 8: Observation of moving average of gradient norms over settings, where $(n_1 = m_2 = r)$, s denotes scaling factors. **(a,b)** illustrate gradient norms when (m_1, n_1) changes from $(4, 4)$ to $(16, 16)$ and $(16, 16)$ to $(64, 64)$, respectively. The rule in Theorem 3.3 is closely approximated. **(c)** The gradients of the empirical optimal training configurations for each rank. Higher parameter counts tend to lower the gradient norms. This further illustrates the point distinguishing between stability and feature learning.

The asymmetry between the two axes clearly shows that stability is governed by n_1 alone, not by m_2, which corroborates the structural interpretation of Theorem 3.3. That is, the $n_1^{1/4}$ factor compensates

for the growth in the operator norm of $\boldsymbol{P}_l^\top \mathbf{L} \boldsymbol{P}_r \mathbf{R}$ as the number of right-factor blocks increases, while m_2 affects only the rank of the adapter (Theorem 3.1) and not its initialization scale.

We also provide practical observation of gradient norms in Fig. 8. The rule for the scaling factor suggests a means to stabilize gradient norms at close values when n_1 varies. However, as noted the limitation discussion in Appendix D.2, while this theoretical rule strictly aligns the gradient scales, it acts primarily as a mathematical safeguard. In optimal empirical scenarios, strictly equating the gradient norm is not always desirable.

D.5 SaMA Converges to the FFT Solution

Convergence to a full fine-tuning solution. To understand the solution quality of SaMA, we analyze a canonical linear least-squares (LSQ) setting. Given input $\mathbf{X} \in \mathbb{R}^{b\times N}$, target $\mathbf{Z} \in \mathbb{R}^{b\times M}$, and frozen weights $\mathbf{W}_0$, the fine-tuning objective is:

$$\min_{\Delta \mathbf{W}} \frac{1}{2} \left\| \mathbf{Z} - \mathbf{X}\mathbf{W}_0^\top - \mathbf{X}\Delta\mathbf{W}^\top \right\|_F^2. \tag{52}$$

Theorem D.1. *[Sparsity] Given any generic rank-ρ matrix $\boldsymbol{X}$, if SaMA configuration: $n_1 \geq \rho$, then any corresponding solution $\Delta\mathbf{W}_{\mathrm{SaMA}}$ to the LSQ problem also solves the full-LSQ in Eq. (52).*

The theorem asserts that, whenever $n_1 \geq \mathrm{rank}(\mathbf{X})$, the SaMA-constrained LSQ problem admits solutions that are simultaneously optimal for the unconstrained (FFT) problem. We prove this by reducing the block-diagonal structure of SaMA to a collection of independent standard LSQ problems, then showing that their closed-form solutions produce a zero gradient in the full unconstrained objective.

Proof Setup

Within the scope of this theorem, $\mathbf{X} \in \mathbb{R}^{b\times N}$ and $\mathbf{Z} \in \mathbb{R}^{b\times M}$ are batches of row vectors. We optimize the Morpho factors $(\mathbf{L}, \mathbf{R})$ directly, absorbing the scaling s into the initialization:

$$\min_{\Delta\mathbf{W}\in\mathcal{F}} \frac{1}{2} \left\| \mathbf{Z} - \mathbf{X}\mathbf{W}_0^\top - \mathbf{X}\Delta\mathbf{W}^\top \right\|_F^2 \equiv \min_{(\mathbf{L},\mathbf{R})\in\mathcal{F}_g} \frac{1}{2} \left\| \mathbf{Z} - \mathbf{X}\mathbf{W}_0^\top - \mathbf{X}\mathbf{R}^\top \boldsymbol{P}_r^\top \mathbf{L}^\top \boldsymbol{P}_l \right\|_F^2. \tag{53}$$

Fix a generic $\mathbf{R}_t \coloneqq \mathbf{R}^\top, \mathrm{rank}(\mathbf{R}_t) = n_1 m_2$ and introduce the substitutions:

$$\mathbf{Y} \coloneqq (\mathbf{Z} - \mathbf{X}\mathbf{W}_0^\top)\boldsymbol{P}_l^\top \in \mathbb{R}^{b\times M}, \qquad \mathbf{T} \coloneqq \mathbf{X}\mathbf{R}_t\boldsymbol{P}_r^\top \in \mathbb{R}^{b\times Q}, \qquad \mathbf{L}_t \coloneqq \mathbf{L}^\top \in \mathbb{R}^{Q\times M}, \tag{54}$$

where $Q = n_1 m_2$. The problem in Eq. (53) reduces to:

$$\min_{\mathbf{L}_t \in \mathcal{BD}} \frac{1}{2} \left\| \mathbf{Y} - \mathbf{T}\mathbf{L}_t \right\|_F^2. \tag{55}$$

Step 1: Block Decomposition into Independent Subproblems

Since $\mathbf{L}_t = \mathrm{diag}(\mathbf{L}_1, \ldots, \mathbf{L}_{m_2})$ with $\mathbf{L}_j \in \mathbb{R}^{n_1\times m_1}$, partition all matrices column-wise into m_2 vertical slices:

$$\mathbf{Y} = [\mathbf{Y}_1 \quad \cdots \quad \mathbf{Y}_{m_2}], \quad \mathbf{T} = [\mathbf{T}_1 \quad \cdots \quad \mathbf{T}_{m_2}], \quad \mathbf{Y}_j \in \mathbb{R}^{b\times m_1},\ \mathbf{T}_j \in \mathbb{R}^{b\times n_1}. \tag{56}$$

The coupled problem in Eq. (55) decouples exactly into m_2 independent standard LSQ problems:

$$\min_{\mathbf{L}_j} \frac{1}{2} \left\| \mathbf{Y}_j - \mathbf{T}_j\mathbf{L}_j \right\|_F^2, \qquad j = 1, \ldots, m_2, \tag{57}$$

each admitting the closed-form minimizer [94]:

$$\mathbf{L}_j^* = \mathbf{T}_j^\dagger \mathbf{Y}_j. \tag{58}$$

Assembling these gives $\mathbf{L}_t^* = \mathrm{diag}(\mathbf{L}_1^*, \ldots, \mathbf{L}_{m_2}^*)$, the solution to Eq. (55).

Step 2 : The Assembled Solution is a Global Minimizer of the Full LSQ

Define $\Delta\mathbf{W}^* \coloneqq \mathbf{R}_t\boldsymbol{P}_r^\top \mathbf{L}_t^* \boldsymbol{P}_l$. The gradient of the full unconstrained objective with respect to $\Delta\mathbf{W}^\top$ is:

$$\mathbf{g}_{\Delta\mathbf{W}^*} = -\mathbf{X}^\top(\mathbf{Z}' - \mathbf{X}\mathbf{R}_t\boldsymbol{P}_r^\top \mathbf{L}_t^* \boldsymbol{P}_l) = -\mathbf{X}^\top(\mathbf{Y} - \mathbf{T}\mathbf{L}_t^*)\boldsymbol{P}_l, \tag{59}$$

where $\mathbf{Z}' = \mathbf{Z} - \mathbf{X}\mathbf{W}_0^\top$. It suffices to show $\mathbf{G}_\Delta \coloneqq -\mathbf{X}^\top(\mathbf{Y} - \mathbf{T}\mathbf{L}_t^*) = \mathbf{0}_{N\times M}$.

Rank factorization of T. Let $\rho = \mathrm{rank}(\mathbf{X})$. Since $\boldsymbol{P}_r^\top$ is orthogonal (full rank) and $\mathrm{rank}(\mathbf{R}_t) = n_1 m_2$ by Theorem 3.1, the rank of $\mathbf{T}$ is determined by the bottleneck:

$$\mathrm{rank}(\mathbf{T}) = \mathrm{rank}(\mathbf{X}\mathbf{R}_t\boldsymbol{P}_r^\top) = \mathrm{rank}(\mathbf{X}) = \rho. \tag{60}$$

By SVD, write $\mathbf{T} = \mathbf{O}\mathbf{Q}$ where $\mathbf{O} \in \mathbb{R}^{b\times\rho}$ has orthonormal columns ($\mathbf{O}^\top\mathbf{O} = \boldsymbol{I}_\rho$, so $\mathbf{O}^\dagger = \mathbf{O}^\top$) and $\mathbf{Q} \in \mathbb{R}^{\rho\times Q}$ has orthonormal rows. Slice column-wise: $\mathbf{T}_j = \mathbf{O}\mathbf{Q}_j$, $\mathbf{Q}_j \in \mathbb{R}^{\rho\times n_1}$. Since $\rho \le n_1$ by assumption, $\mathbf{Q}_j$ is full row-rank due to generic matrices, giving:

$$\mathbf{Q}_j\mathbf{Q}_j^\dagger = \boldsymbol{I}_\rho. \tag{61}$$

Because $\mathbf{O}$ is full column-rank and $\mathbf{Q}_j$ is full row-rank, the pseudoinverse distributes over their product [95]:

$$\mathbf{T}_j^\dagger = (\mathbf{O}\mathbf{Q}_j)^\dagger = \mathbf{Q}_j^\dagger\mathbf{O}^\top. \tag{62}$$

Gradient vanishes blockwise. Partition $\mathbf{X}$ into n_2 horizontal slices $\mathbf{X}_i \in \mathbb{R}^{b\times n_1}$. The (i,j)-th block of $\mathbf{G}_\Delta$ is:

$$\begin{aligned}
\mathbf{G}_{\Delta_{i,j}} &= \mathbf{X}_i^\top\mathbf{T}_j\mathbf{L}_j^* - \mathbf{X}_i^\top\mathbf{Y}_j \\
&= \mathbf{X}_i^\top(\mathbf{O}\mathbf{Q}_j)(\mathbf{Q}_j^\dagger\mathbf{O}^\top)\mathbf{Y}_j - \mathbf{X}_i^\top\mathbf{Y}_j \\
&= \mathbf{X}_i^\top\mathbf{O}(\mathbf{Q}_j\mathbf{Q}_j^\dagger)\mathbf{O}^\top\mathbf{Y}_j - \mathbf{X}_i^\top\mathbf{Y}_j \\
&= \mathbf{X}_i^\top\mathbf{O}\boldsymbol{I}_\rho\mathbf{O}^\top\mathbf{Y}_j - \mathbf{X}_i^\top\mathbf{Y}_j \\
&= \mathbf{X}_i^\top(\mathbf{O}\mathbf{O}^\top)\mathbf{Y}_j - \mathbf{X}_i^\top\mathbf{Y}_j.
\end{aligned} \tag{63}$$

Since $\mathbf{O}$ has orthonormal columns spanning $\mathrm{col}(\mathbf{T}) = \mathrm{col}(\mathbf{X})$, the projection $\mathbf{O}\mathbf{O}^\top$ is the orthogonal projector onto $\mathrm{col}(\mathbf{X})$. Because $\mathbf{X}_i^\top$ maps into $\mathrm{col}(\mathbf{X})^\perp$'s annihilator, $\mathbf{X}_i^\top\mathbf{O}\mathbf{O}^\top = \mathbf{X}_i^\top$, and thus:

$$\mathbf{G}_{\Delta_{i,j}} = \mathbf{X}_i^\top\mathbf{Y}_j - \mathbf{X}_i^\top\mathbf{Y}_j = \mathbf{0}_{n_1\times m_1}, \qquad \forall\, i,j. \tag{64}$$

Hence $\mathbf{G}_\Delta = \mathbf{0}_{N\times M}$, and $\Delta\mathbf{W}^*$ is a stationary point— and therefore a global minimizer—of the full LSQ objective.

Step 3 : Extension to the Full Solution Set

The argument above uses the minimum-norm solution $\mathbf{L}_j^* = \mathbf{T}_j^\dagger\mathbf{Y}_j$. When $\rho < n_1$, the system $\mathbf{T}_j\mathbf{L}_j = \mathbf{Y}_j$ is underdetermined and the complete solution set is [94]:

$$\mathbf{L}_j^* = \mathbf{T}_j^\dagger\mathbf{Y}_j + (\boldsymbol{I}_{n_1} - \mathbf{T}_j^\dagger\mathbf{T}_j)\boldsymbol{U}, \qquad \boldsymbol{U} \in \mathbb{R}^{n_1\times m_1}. \tag{65}$$

The null-space term drops out of the gradient:

$$\begin{aligned}
\mathbf{X}_i^\top\mathbf{T}_j\mathbf{L}_j^* &= \mathbf{X}_i^\top\mathbf{Y}_j + \mathbf{X}_i^\top\mathbf{T}_j(\boldsymbol{I}_{n_1} - \mathbf{T}_j^\dagger\mathbf{T}_j)\boldsymbol{U} \\
&= \mathbf{X}_i^\top\mathbf{Y}_j + \mathbf{X}_i^\top(\mathbf{T}_j - \mathbf{T}_j\mathbf{T}_j^\dagger\mathbf{T}_j)\boldsymbol{U} \\
&= \mathbf{X}_i^\top\mathbf{Y}_j,
\end{aligned} \tag{66}$$

using $\mathbf{T}_j\mathbf{T}_j^\dagger\mathbf{T}_j = \mathbf{T}_j$. The gradient vanishes identically across the entire solution set. □

Empirical Verification

Figure 9 confirms the gradient alignment predicted by Theorem D.1: when $n_1 \ge \mathrm{rank}(\mathbf{X})$, Morpho gradient norms track those of FFT throughout training. Figure 10 reveals an important nuance: point-wise precision (OCS) and subspace alignment (ASA) degrade at *different* rank thresholds. OCS collapses at n_1—the local block capacity—while ASA remains near 1.0 until the global adapter rank $n_1 m_2$. This separation demonstrates that Morpho maintains the correct geometric subspace of the target transformation long after exact numerical reconstruction becomes infeasible, a property not captured by point-wise metrics alone.

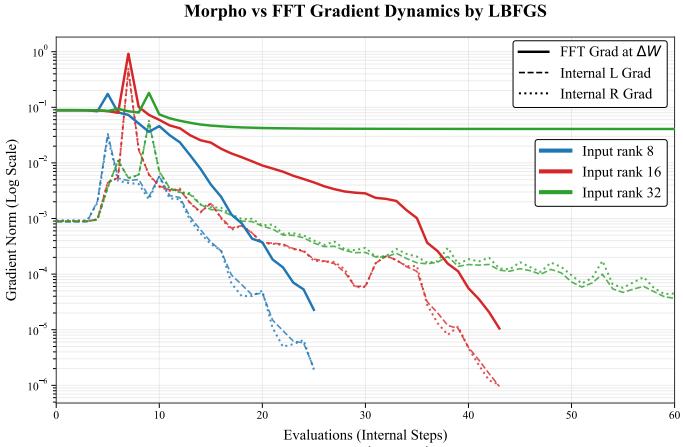

Figure 9: Gradient dynamics of Morpho factors $(\mathbf{L}, \mathbf{R})$ and the FFT baseline over training, at adapter ranks $\{8, 16, 32\}$. When $n_1 \geq \mathrm{rank}(\mathbf{X})$, the Morpho gradient norm tracks the FFT gradient norm closely throughout optimization, confirming Theorem D.1. Divergence emerges only when $n_1 < \mathrm{rank}(\mathbf{X})$, marking the rank threshold predicted by the theorem.

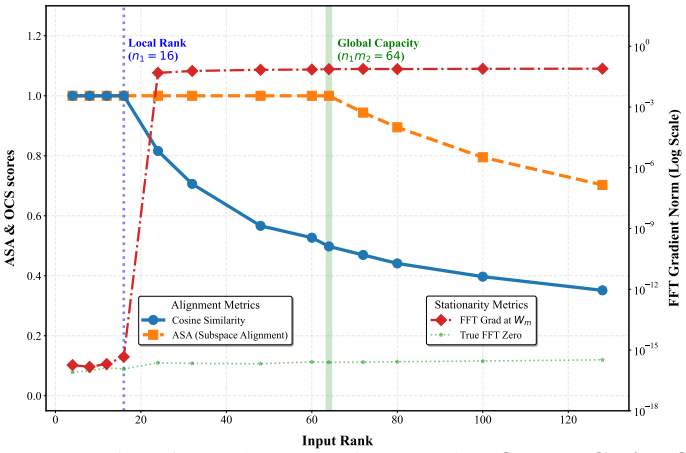

Figure 10: Structural capacity of Morpho across input ranks. **Output Cosine Similarity (OCS)** measures point-wise directional alignment between Morpho and FFT outputs—a precision metric for exact numerical reconstruction. **Averaged Subspace Alignment (ASA)** measures how well the column space of $\Delta\mathbf{W}_{\mathrm{SaMA}}$ covers that of $\Delta\mathbf{W}_{\mathrm{FFT}}$—a coarser but more robust structural metric. Two critical thresholds emerge: at n_1 (the left-factor rank), OCS collapses as individual blocks lose sufficient degrees of freedom for exact reconstruction; at $n_1 m_2$ (the Morpho adapter rank), ASA begins to gradually decay from 1.0, showing that Morpho preserves the fundamental geometric subspace of the target transformation well beyond the point-wise precision threshold.

D.6 Gauge Invariance in PEFT: A Unified Perspective and the Parameter-Space Uniqueness of Morpho

A central but underappreciated obstacle in factorized fine-tuning is *gauge invariance* which is known as the existence of a continuous family of internal parameterizations that all produce the identical adapter matrix. It induces flat directions in the loss landscape, destabilizes gradient-based optimizers. We first expose this phenomenon across representative PEFT families, revealing it as a pervasive structural defect rather than an isolated LoRA artifact. We then prove that the invariant manifold of Morpho parameterization reduces to *diagonal scalings*.

D.6.1 Gauge Invariance: A Taxonomy of Failure Modes

We unify the analysis under a single framework. Let $\Phi(\boldsymbol{\theta})$ denote any parameterized adapter, mapping a tuple of trainable tensors $\boldsymbol{\theta}$ to an update matrix $\Delta\mathbf{W}$. We say Φ admits a *gauge group* $\mathcal{G}$ if there exists a non-trivial group action $g \cdot \boldsymbol{\theta} \mapsto \boldsymbol{\theta}'$ such that $\Phi(\boldsymbol{\theta}') = \Phi(\boldsymbol{\theta})$ for all $g \in \mathcal{G}$. The larger $\mathcal{G}$ is, the more severe the redundancy becomes.

LoRA: $\mathcal{G} = \mathrm{GL}(r)$. LoRA [7] writes $\Delta\mathbf{W} = \mathbf{BA}$, $\mathbf{B} \in \mathbb{R}^{M\times r}$, $\mathbf{A} \in \mathbb{R}^{r\times N}$. For any invertible $\mathbf{C} \in \mathbb{R}^{r\times r}$:

$$\Delta\mathbf{W} = \mathbf{BA} = (\mathbf{BC})(\mathbf{C}^{-1}\mathbf{A}) = \mathbf{B}'\mathbf{A}'. \tag{67}$$

The gauge group is the full general linear group $\mathrm{GL}(r)$, which is r^2-dimensional. Every point in parameter space lies at the intersection of an r^2-dimensional flat valley. This symmetry is broken only at the boundary of the feasible set (e.g. when $\mathbf{A}$ or $\mathbf{B}$ becomes rank-deficient), which is measure-zero but attracting for gradient descent near convergence, producing the well-documented *rank collapse* phenomenon [16]. The same $\mathrm{GL}(r)$ redundancy propagates verbatim to DoRA [28], PiSSA [8], MiLoRA [64], HiRA [19], RandLoRA [51], and ABBA [35], since all of them retain the $\mathbf{BA}$ formulation at their core.

QuanTA: Multi-linear $\mathcal{G} = \mathrm{GL}(d_1) \times \cdots \times \mathrm{GL}(d_K)$. QuanTA [11] contracts a chain of parameter tensors $\{\mathcal{T}_k\}_{k=1}^{K}$ over shared latent dimensions $\{d_k\}$. The weight update (suppressing input indices) is:

$$\Delta\mathcal{W}_{g,h,i,a,b,c} = \sum_{d,e,f} (\mathcal{T}_1)_{g,h,d,e}\,(\mathcal{T}_2)_{d,i,a,f}\,(\mathcal{T}_3)_{e,f,b,c}. \tag{68}$$

Inserting identity resolutions $\mathbf{C}_d^{-1}\mathbf{C}_d = \boldsymbol{I}$ along each shared latent dimension d independently:

$$\mathcal{T}_1 \times_d \mathcal{T}_2 = (\mathcal{T}_1 \times_d \mathbf{C}_d^{-1}) \times_d (\mathbf{C}_d \times_d \mathcal{T}_2) = \tilde{\mathcal{T}}_1 \times_d \tilde{\mathcal{T}}_2, \qquad \mathbf{C}_d \in \mathrm{GL}(d). \tag{69}$$

Since this freedom is independent across every shared axis, the total gauge group is $\prod_k \mathrm{GL}(d_k)$, whose dimension is $\sum_k d_k^2$. For typical QuanTA configurations with $K = 3$ and $d_k \sim 8$–64, this amounts to hundreds of redundant degrees of freedom per adapter, making the loss landscape exponentially more degenerate than LoRA.

Orthogonal fine-tuning (OFT / BOFT): $\mathcal{G} = O(r)$. Orthogonal Fine-Tuning [59] and its butterfly variant BOFT [96] parameterize the adapter as an orthogonal matrix $\mathbf{Q}$, updating the weight as $\mathbf{W}_{ft} = \mathbf{QW}_0$. Because $\mathbf{Q}$ is constrained to $O(N)$, a direct dense gauge group is not available; however, within the block structure of BOFT, each block admits an $O(b)$ ambiguity (b being the block size) in its parameterization via skew-symmetric generators, yielding a gauge group $\prod_k O(b)$. For skew-symmetric parameterizations of the form $\mathbf{A} = \mathbf{UV}^\top - \mathbf{VU}^\top$, the substitution $\mathbf{U} \leftarrow \mathbf{UC}$, $\mathbf{V} \leftarrow \mathbf{VC}$ for any $\mathbf{C} \in O(r)$ leaves $\mathbf{A}$ unchanged, confirming an $O(r)$-valued gauge symmetry.

Summary. Across the PEFT landscape, gauge invariance is the rule, not the exception. The severity is determined by the dimensionality of the gauge group: $\dim \mathcal{G} = r^2$ for LoRA; $\sum d_k^2$ for QuanTA; $\binom{r}{2}$ for orthogonal methods.

D.6.2 Invariant Manifold of Morpho Parameterization

We build the theory in three steps as follows:

Step 1: The Kronecker Baseline We first establish uniqueness for the most structured case—a pure Kronecker product—as the theoretical bedrock.

Theorem D.2 (Uniqueness of Kronecker factorization)**.** *Let* $\mathbf{K} = \mathbf{A}_1 \otimes \mathbf{B}_1 \neq \mathbf{0}$*, with* $\mathbf{A}_1 \in \mathbb{R}^{m_1 \times n_1}$, $\mathbf{B}_1 \in \mathbb{R}^{m_2 \times n_2}$*. If* $\mathbf{C} \otimes \mathbf{D} = \mathbf{K}$ *for some* $\mathbf{C} \in \mathbb{R}^{m_1 \times n_1}$, $\mathbf{D} \in \mathbb{R}^{m_2 \times n_2}$*, then:*

$$\exists\, k \in \tilde{\mathbb{R}} : \quad \mathbf{C} = k^{-1}\mathbf{A}_1, \quad \mathbf{D} = k\,\mathbf{B}_1. \tag{70}$$

Proof. Since $\mathbf{K} \neq \mathbf{0}$, at least one block entry $a_{ij}\mathbf{B}_1 \neq \mathbf{0}$ of the first Kronecker expansion equals $c_{ij}\mathbf{D}$ of the second. With $a_{ij}, c_{ij} \neq 0$ (otherwise the block is zero in both), it follows that $\mathbf{D} = k\,\mathbf{B}_1$ where $k = a_{ij}/c_{ij}$. Substituting back and applying the cancellation law for Kronecker products:

$$\mathbf{A}_1 \otimes \mathbf{B}_1 = \mathbf{C} \otimes (k\,\mathbf{B}_1) = (k\,\mathbf{C}) \otimes \mathbf{B}_1 \quad \Rightarrow \quad \mathbf{A}_1 = k\,\mathbf{C}. \tag{71}$$

Hence $\mathbf{C} = k^{-1}\mathbf{A}_1$, and the scalar k is the sole degree of freedom. □

Theorem D.2 establishes a striking contrast with LoRA: the Kronecker product $\mathbf{A} \otimes \mathbf{B}$ uniquely determines the pair $(\mathbf{A}, \mathbf{B})$ up to a *single scalar*. The gauge group is $\mathcal{G} = \tilde{\mathbb{R}} \cong (\tilde{\mathbb{R}})^1$, reducing r^2 degrees of freedom to exactly 1.

Step 2: Square Monarch Matrices Monarch parameterization extends Kronecker products to square matrices via two block-diagonal factors and a permutation. We characterize its full invariant manifold for a N-by-N square matrix, factorized by $n : n^2 = N$

Lemma D.7 (Permutation conjugation of diagonal matrices)**.** *Let* $\boldsymbol{D}_L = \mathrm{diag}(\boldsymbol{v})$ *be an invertible diagonal matrix and* $\boldsymbol{P}_1 = \boldsymbol{P}_{n,n}$ *the perfect shuffle permutation. Define* $\mathbf{L} = \mathbf{L}_0\boldsymbol{D}_L$ *and* $\mathbf{R} = \boldsymbol{P}_1^\top \boldsymbol{D}_L^{-1}\boldsymbol{P}_1\mathbf{R}_0$*. Then* $(\mathbf{L}, \mathbf{R})$ *are block-diagonal with the same block structure as* $(\mathbf{L}_0, \mathbf{R}_0)$*.*

Proof. Right-multiplying a block-diagonal matrix $\mathbf{L}_0$ by a diagonal $\boldsymbol{D}_L$ scales each row independently, preserving the block structure: $\mathbf{L}_{ij} = (\mathbf{L}_0)_{ij}(D_L)_{jj}$. For $\mathbf{R}$, observe that $\boldsymbol{Q} := \boldsymbol{P}_1^\top \boldsymbol{D}_L^{-1}\boldsymbol{P}_1$ satisfies $\boldsymbol{Q}_{ij} = d_{\pi(j)}[e_{\pi(i)}^\top e_{\pi(j)}]$, which is zero for $i \neq j$ and equals $d_{\pi(j)}$ for $i = j$. Thus $\boldsymbol{Q}$ is diagonal with entries permuted by π, and left-multiplying a block-diagonal $\mathbf{R}_0$ by a diagonal matrix preserves its block structure. □

Lemma D.8 (Reverse direction: block-diagonality forces diagonality)**.** *Let* $\boldsymbol{D}_L, \boldsymbol{D}_R$ *satisfy* $\mathbf{L} = \mathbf{L}_0\boldsymbol{D}_L$ *and* $\mathbf{R} = \boldsymbol{D}_R^{-1}\mathbf{R}_0$ *with* $(\mathbf{L}, \mathbf{R}, \mathbf{L}_0, \mathbf{R}_0) \in \mathcal{BD}^{(n)}$ *(block-diagonal with* $n \times n$ *blocks), and* $\boldsymbol{D}_R = \boldsymbol{P}_1^\top \boldsymbol{D}_L \boldsymbol{P}_1$*. Then* $\boldsymbol{D}_L$ *and* $\boldsymbol{D}_R$ *are diagonal.*

Proof. Since $(\mathbf{L}, \mathbf{L}_0)$ and $(\mathbf{R}, \mathbf{R}_0)$ are pairwise block-diagonal, $\boldsymbol{D}_L, \boldsymbol{D}_R \in \mathcal{BD}$. Write the permutation function π of $\boldsymbol{P}_{n,n}$ explicitly (1-indexed):

$$\pi : i \mapsto n\big((i-1) \bmod n\big) + \lceil i/n \rceil. \tag{72}$$

The (i,j)-entry of $\boldsymbol{D}_R = \boldsymbol{P}_1^\top \boldsymbol{D}_L \boldsymbol{P}_1$ is $(\boldsymbol{D}_L)_{\pi(i),\pi(j)}$. For a principal entry (i,j) within the k-th diagonal block of $\boldsymbol{D}_R$ (so $(k-1)n < i, j \leq kn$ or $\lceil i/n \rceil = \lceil j/n \rceil = k$), write $i = (k-1)n + l_i$, $j = (k-1)n + l_j$ with $l_i, l_j \in \overline{1..n}$. The relocated index satisfies $|i-j|_{\boldsymbol{D}_R} = |l_i - l_j| n \geq n$ whenever $i \neq j$, placing it strictly outside any diagonal block of $\boldsymbol{D}_L$. Since $\boldsymbol{D}_L \in \mathcal{BD}$ requires off-diagonal-block entries to vanish, we conclude $(\boldsymbol{D}_R)_{ij} = 0$ for all $i \neq j$, i.e. $\boldsymbol{D}_R$ is diagonal, and then so is $\boldsymbol{D}_L$ by the same argument applied in reverse. □

Theorem D.3 (Invariant manifold of square Monarch / Morpho)**.** *Let* $\mathbf{M}_0 = \boldsymbol{P}_1^\top \mathbf{L}_0 \boldsymbol{P}_1 \mathbf{R}_0 \in \mathcal{F}^{(N)}$ *be a square Monarch matrix. The set of all generative factor pairs is:*

$$\mathcal{F}_g(\mathbf{M}_0) = \Big\{ (\mathbf{L}, \mathbf{R}) \;\Big|\; \mathbf{L} = \mathbf{L}_0\boldsymbol{D},\ \mathbf{R} = \boldsymbol{P}_1^\top \boldsymbol{D}^{-1}\boldsymbol{P}_1\mathbf{R}_0;\ \boldsymbol{D} = \mathrm{diag}(\boldsymbol{v}),\ \boldsymbol{v} \in \tilde{\mathbb{R}}^N \Big\}. \tag{73}$$

The gauge group is isomorphic to $\tilde{\mathbb{R}}^N$*, the* N*-dimensional multiplicative torus.*

Proof. Let $(\mathbf{L}, \mathbf{R}) \in \mathcal{F}_g(\mathbf{M}_0)$, so $\boldsymbol{P}_1^\top \mathbf{L} \boldsymbol{P}_1 \mathbf{R} = \boldsymbol{P}_1^\top \mathbf{L}_0 \boldsymbol{P}_1 \mathbf{R}_0$. Cancelling the leading $\boldsymbol{P}_1^\top$ and right-multiplying by $\mathbf{R}^{-1}$ (which exists almost surely in the random-initialization regime; see Appendix D.1):

$$\mathbf{L}_0^{-1}\mathbf{L}\boldsymbol{P}_1 = \boldsymbol{P}_1\mathbf{R}_0\mathbf{R}^{-1} \quad \Rightarrow \quad \boldsymbol{D}_L\boldsymbol{P}_1 = \boldsymbol{P}_1\boldsymbol{D}_R, \tag{74}$$

where $\boldsymbol{D}_L = \mathbf{L}_0^{-1}\mathbf{L}$ and $\boldsymbol{D}_R = \mathbf{R}_0\mathbf{R}^{-1}$, giving the conjugation relation $\boldsymbol{D}_R = \boldsymbol{P}_1^\top \boldsymbol{D}_L \boldsymbol{P}_1$. By Lemma D.8, $\boldsymbol{D}_L$ is diagonal; by Lemma D.7, $(\mathbf{L}, \mathbf{R})$ are valid block-diagonal factors. The forward direction (any diagonal $\boldsymbol{D}$ yields a valid reparameterization) is immediate from Lemma D.7. □

Step 3: General Morpho (Rectangle)

Lemma D.9 (Pseudoinverse block-preservation for rectangular factors)**.** *Let $\mathbf{L} \in \mathbb{R}^{M\times q}$ be full column-rank block-diagonal and $\mathbf{R} \in \mathbb{R}^{q\times N}$ be full row-rank block-diagonal with $q = n_1 m_2 < \min(M, N)$. If $\mathbf{L} = \mathbf{L}_1 \boldsymbol{D}_{Lr}$ and $\mathbf{R} = \boldsymbol{D}_{Rr}^{-1}\mathbf{R}_1$ with $\boldsymbol{D}_{Rr} = \boldsymbol{P}_r^\top \boldsymbol{D}_{Lr}\boldsymbol{P}_r$, then $\boldsymbol{D}_{Lr}$ and $\boldsymbol{D}_{Rr}$ are diagonal.*

Proof. Since block-diagonal matrices with full-rank blocks admit pseudoinverses that decouple blockwise (Moore–Penrose uniqueness [97]), both $\boldsymbol{D}_{Lr} = \mathbf{L}_1^\dagger \mathbf{L}$ and $\boldsymbol{D}_{Rr} = \mathbf{R}_1\mathbf{R}^\dagger$ lie in $\mathcal{BD}$, specifically as $q \times q$ square matrices with m_2 blocks of size $n_1 \times n_1$ and n_1 blocks of size $m_2 \times m_2$, respectively. The conjugation $\boldsymbol{D}_{Rr} = \boldsymbol{P}_r^\top \boldsymbol{D}_{Lr}\boldsymbol{P}_r$ with permutation $\boldsymbol{P}_r = \boldsymbol{P}_{n_1,m_2}$ (with $\pi(i) = \pi_{n_1,m_2}(i) = n_1\,[(i-1) \bmod m_2] + \lceil i/m_2 \rceil$) relocates each off-diagonal principal entry of $\boldsymbol{D}_{Lr}$ to a position outside any block of $\boldsymbol{D}_{Rr}$. Since $\boldsymbol{D}_{Rr} \in \mathcal{BD}$ forces those entries to zero, $\boldsymbol{D}_{Lr}$ must be diagonal, and hence so is $\boldsymbol{D}_{Rr}$. □

Theorem D.4 (Gauge invariance of parameter-efficient Morpho)**.** *Let $\mathbf{M}_1 = \boldsymbol{P}_l^\top \mathbf{L}_1 \boldsymbol{P}_r \mathbf{R}_1 \in \mathcal{F}^{(M,N)}$ with hidden dimension $q = n_1 m_2 \le \min(M, N)$. The invariant manifold of generative factor pairs is:*

$$\mathcal{F}_g(\mathbf{M}_1) = \left\{ (\mathbf{L}, \mathbf{R}) \;\middle|\; \mathbf{L} = \mathbf{L}_1 \boldsymbol{D},\ \mathbf{R} = \boldsymbol{P}_r^\top \boldsymbol{D}^{-1} \boldsymbol{P}_r \mathbf{R}_1;\ \boldsymbol{D} = \mathrm{diag}(\boldsymbol{v}),\ \boldsymbol{v} \in \tilde{\mathbb{R}}^q \right\}. \tag{75}$$

The gauge group is $\mathcal{G}_{Morpho} \cong (\tilde{\mathbb{R}})^q$.

Proof. For a randomly initialized adapter, $\mathbf{L}_1$ is full column-rank and $\mathbf{R}_1$ is full row-rank almost surely. From $\boldsymbol{P}_l^\top \mathbf{L}\boldsymbol{P}_r\mathbf{R} = \boldsymbol{P}_l^\top \mathbf{L}_1 \boldsymbol{P}_r \mathbf{R}_1$, left-multiply by $\mathbf{L}_1^\dagger \boldsymbol{P}_l$ and right-multiply by $\mathbf{R}^\dagger$:

$$\boldsymbol{D}_{L1}\boldsymbol{P}_r = \boldsymbol{P}_r \boldsymbol{D}_{R1}, \quad \boldsymbol{D}_{L1} = \mathbf{L}_1^\dagger \mathbf{L}, \quad \boldsymbol{D}_{R1} = \mathbf{R}_1 \mathbf{R}^\dagger, \tag{76}$$

yielding the conjugation $\boldsymbol{D}_{R1} = \boldsymbol{P}_r^\top \boldsymbol{D}_{L1} \boldsymbol{P}_r$. By Lemma D.9, $\boldsymbol{D}_{L1}$ is diagonal; by Lemma D.7, any diagonal $\boldsymbol{D}$ generates a valid block-diagonal reparameterization. □

D.6.3 Discussion: Why Diagonal Gauge Structure Matters for Optimization

The geometry of degeneracy. Theorem D.4 establishes that the invariant manifold of Morpho is a q-dimensional torus denoted as $\mathcal{T}^q = (\tilde{\mathbb{R}})^q$. In contrast, LoRA's gauge group $\mathrm{GL}(r)$ contains not only diagonal scalings but also rotations, shearings, and reflections. All of these are invisible to the loss function yet create curved, degenerate valleys that optimizers may navigate poorly.

Decoupling magnitude from direction. The $\mathrm{GL}(r)$ symmetry of LoRA conflates magnitude and directional updates in the parameter space. DoRA [28] addresses this empirically by explicitly decomposing weight updates into magnitude and direction components. Morpho resolves the same issue structurally: since the only reparameterization freedom is diagonal scaling, the i-th hidden feature's magnitude is encoded by $\boldsymbol{D}_{ii}$, while its direction is fixed by $\mathbf{L}_i$ and $\mathbf{R}_i$. Magnitude and direction are *structurally decoupled* without requiring any post-hoc decomposition.

D.7 Distinction from MoRE Fine-tuning

Although both SaMA and MoRE [30] leverage perfect shuffle permutations in their designs, they differ fundamentally in their algebraic factorization, resulting in distinct interaction topologies.

Consider a MoRE adapter $\mathbf{W} \in \mathbb{R}^{M\times N}$, defined as $\mathbf{W} = \boldsymbol{P}_2 \mathbf{L}_B \cdot \boldsymbol{P}_z \cdot \mathbf{R}_B$, where b is the number of blocks in both $\mathbf{L}_B$ and $\mathbf{R}_B$, r_{blk} is the local rank of each block, and $\boldsymbol{P}_z, \boldsymbol{P}_2$ are the stride permutation matrices as prescribed in the MoRE configuration. The matrices $\mathbf{L}_B$ and $\mathbf{R}_B$ consist of submatrices of size $m \times r_{blk}$ and $r_{blk} \times n$, respectively, where $m = M/b$ and $n = N/b$. In the context of reparameterization, we omit the final permutation matrix $\mathbf{P}_2$ in the analysis and simply write: $\mathbf{W} = \mathbf{L}_B \cdot \boldsymbol{P}_z \cdot \mathbf{R}_B$, as it cancels out when comparing two adapters of the same configuration.

$$\begin{aligned} \boldsymbol{P}_z &= \boldsymbol{P}_{r_{blk},b}, \\ \mathbf{L}_B = \mathrm{diag}(\mathbf{L}_1, \ldots, \mathbf{L}_b), &\quad \mathbf{L}_i \in \mathbb{R}^{m\times r_{blk}}, 1 \le i \le b, \\ \mathbf{R}_B = \mathrm{diag}(\mathbf{R}_1, \ldots, \mathbf{R}_b), &\quad \mathbf{R}_j \in \mathbb{R}^{r_{blk}\times n}, 1 \le j \le b. \end{aligned}$$

D.7.1 Different Objectives in Structured Matrix Design

Core Motivation: MoRE prioritizes sparsity preservation to minimize parameters, whereas SaMA extends Kronecker factorization to gain flexibility, scalability, and global interactions.

Architectural Principle: MoRE follows a partitioned low-rank paradigm: it splits the representation space into isolated, group-wise low-rank blocks and relies on permutations to mix them. In dense adapter regimes, this design suffers from information isolation (Fig. 11). In contrast, SaMA applies a bilateral structured permutation ($\boldsymbol{P}_l^\top \mathbf{L} \boldsymbol{P}_r$) that converts the factors into block-diagonal matrices, yielding a globally distributed interaction grid inherited from the Kronecker product (Fig. 12).

Design Strategy: Consistent with its partitioning principle, MoRE divides both factors into b low-rank blocks, so its parameter count is dictated solely by the block rank r_{blk}. SaMA instead derives the number of blocks per factor from block shapes – a choice rooted in the Kronecker view – which establishes mutual connections across partitions.

Table 6: Summary of fundamental structural and algebraic differences between MoRE and SaMA, where b and r_{blk} denote the number of blocks and the block rank, respectively.

Feature	**MoRE**	**SaMA (Ours)**
Algebraic Form	Multi-LoRA + Permutation	Kronecker Product Generalization
Interaction Topology	Fragmented (Isolated subspaces)	Guaranteed 100% Dense
Factor Shapes	$\mathbf{L}_B : (b, M/b, r_{blk})$ $\mathbf{R}_B : (b, r_{blk}, N/b)$	$\mathbf{L} : (m_2, m_1, n_1)$ $\mathbf{R} : (n_1, m_2, n_2)$
Parameter Count θ	$r_{blk}(M+N)$	$n_1 M + m_2 N$
Invariant Manifold	Allows off-diagonal interactions	Diagonal matrices only

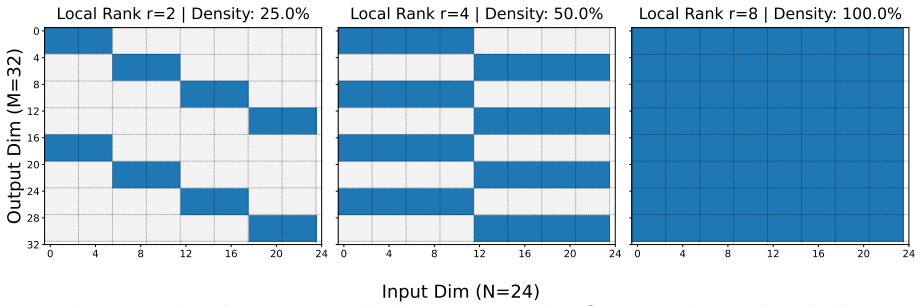

Figure 11: Binary masks of MoRE's equivalent output (when $b = 8$). Blue regions indicate trainable cross-neuron connections. The visualization suggests that MoRE inherently enforces a fragmented, sparse interaction topology when $r_{blk} < b$. Global dense connectivity is exclusively conditionally triggered when the local rank matches the number of blocks ($r_{blk} = b$).

D.7.2 MoRE fails to restrain the invariant manifold to diagonal transformations

Theorem D.5 (Reparameterization in MoRE)**.** *Assume $r_{blk} > b > 1$, there exists a non-diagonal invertible matrix $\boldsymbol{C} \in \mathbb{R}^{br_{blk} \times br_{blk}}$ that reparameterizes the MoRE factors as:*

$$\mathbf{L}'_B = \mathbf{L}_B\,(\boldsymbol{P}_z \boldsymbol{C} \boldsymbol{P}_z^\top), \qquad \mathbf{R}'_B = \boldsymbol{C}^{-1}\mathbf{R}_B,$$

while leaving the resulting weight $\mathbf{W}$ invariant.

Constructive Proof. For any invertible $\boldsymbol{C} \in \mathbb{R}^{br_{blk} \times br_{blk}}$, inserting $\boldsymbol{P}_z \boldsymbol{C} \boldsymbol{P}_z^\top \cdot \boldsymbol{P}_z \boldsymbol{C}^{-1} \boldsymbol{P}_z^\top = \boldsymbol{I}$ between $\mathbf{L}_B$ and $\boldsymbol{P}_z$ gives:

$$\mathbf{W} \;=\; \mathbf{L}_B\, \boldsymbol{P}_z\, \mathbf{R}_B \;=\; \underbrace{\mathbf{L}_B(\boldsymbol{P}_z \boldsymbol{C} \boldsymbol{P}_z^\top)}_{\mathbf{L}'_B}\; \boldsymbol{P}_z\; \underbrace{\boldsymbol{C}^{-1}\mathbf{R}_B}_{\mathbf{R}'_B}.$$

It remains to construct a non-diagonal $\boldsymbol{C}$ such that $(\mathbf{L}'_B, \mathbf{R}'_B)$ are valid MoRE factors, i.e. both retain the prescribed block-diagonal structure.

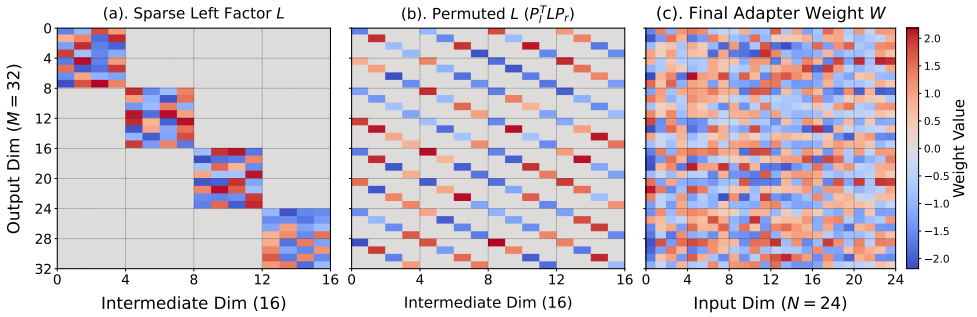

Figure 12: The structural transformation pipeline of SaMA. The heatmaps illustrate the deterministic expansion of the parameterization space. The sparse, block-diagonal left factor (a) is reorganized by modulo-based stride permutations into a tiled sub-diagonal pattern (b), effectively executing a Kronecker operand swap. This scattered representation is then integrated by the right factor to produce a fully dense global weight matrix (c), achieving complete input-output feature mixing regardless of adapter configurations.

Construction. Take $\boldsymbol{C} = \boldsymbol{I} + \alpha\,\mathbf{E}_{u,v}$, with $\alpha \in \tilde{\mathbb{R}}$ and $u \neq v$, where $\mathbf{E}_{u,v}$ has a single 1 at position (u, v). Such $\boldsymbol{C}$ is automatically non-diagonal and invertible (with the inverse $\boldsymbol{C}^{-1} = \boldsymbol{I} - \alpha\mathbf{E}_{u,v}$). Validity then reduces to choosing (u, v) so that the block-sparsity patterns of $\mathbf{R}_B$ and $\mathbf{L}_B$ are preserved:

1. **Right preservation.** $\boldsymbol{C}^{-1}$ acts on $\mathbf{R}_B$ by row mixing. Confining the mixing within a single block of $\mathbf{R}_B$ requires $\lceil u/r_{blk} \rceil = \lceil v/r_{blk} \rceil$.
2. **Left preservation.** The conjugated factor $\boldsymbol{C}_L = \boldsymbol{P}_z \boldsymbol{C} \boldsymbol{P}_z^\top$ acts on $\mathbf{L}_B$ by column mixing. Letting π denote the index permutation induced by $\boldsymbol{P}_z$, hence $\boldsymbol{P}_z \boldsymbol{e}_u = \boldsymbol{e}_{\pi(u)}$, leads to $\boldsymbol{P}_z \mathbf{E}_{u,v} \boldsymbol{P}_z^\top = \mathbf{E}_{\pi(u),\pi v}$, therefore the off-diagonal entry must remain inside one block of $\mathbf{L}_B$ if: $\lceil \pi(u)/r_{blk} \rceil = \lceil \pi(v)/r_{blk} \rceil$.

Existence of valid (u, v). Write an index in $\mathbf{R}_B$ given by $u = (k-1) \cdot b + l$ with $k \in \overline{1..r_{blk}}$ and $l \in \overline{1..b}$. The stride permutation map $u \mapsto \pi(u) = (l-1) \cdot r_{blk} + k$, i.e. u is sent to the k-th position of block l of $\mathbf{L}_B$.

By selecting adjacent coefficients $k_u = 1, k_v = 2$ and offset $(l = 1)$, the linear indices $u = 1$ and $v = b + 1$ both belong to block 1 of $\mathbf{R}_B$ since $r_{blk} > b$. Their images under π are $\pi(u) = 1$ and $\pi(v) = 2$, both mapped indices remain in block 1 of $\mathbf{L}_B$. Thus $(u, v) = (1, b + 1)$ satisfies both preservation conditions.

Verification. By construction, $\boldsymbol{C}$ is non-diagonal and invertible. Condition 1 confines the mixing of $\boldsymbol{C}^{-1}$ to a single block of $\mathbf{R}_B$, so $\mathbf{R}'_B \in \mathcal{BD}$. Condition 2 places the off-diagonal entry of $\boldsymbol{C}_L$ inside one block of $\mathbf{L}_B$, so $\boldsymbol{C}_L \in \mathcal{BD}$ and consequently $\mathbf{L}'_B = \mathbf{L}_B \boldsymbol{C}_L \in \mathcal{BD}$. Hence $(\mathbf{L}'_B, \mathbf{R}'_B)$ is a valid MoRE reparameterization of $\mathbf{W}$. □

The construction generalizes immediately to $\boldsymbol{C} = \boldsymbol{I} + \sum_{(u,v)\in\mathcal{S}} \alpha_{u,v} \mathbf{E}_{u,v}$ for any set $\mathcal{S}$ of pairs satisfying both conditions, provided $\boldsymbol{C}$ is invertible (e.g. $\mathcal{S} = \{(1, b{+}1), (b{+}1, 1)\}$ with $\alpha_{1,b+1}\alpha_{b+1,1} = 1$ yields $\det \boldsymbol{C} = 0$). Since our purpose is only to show that the invariant manifold of MoRE strictly extends beyond the diagonal subspace $\mathcal{D}$, we leave a complete characterization of this extended subspace to future work.

As a direct corollary of Theorem D.5, the MoRE design admits mutual interactions across its invariant manifold, at least when $r_{blk} > b$, which can introduce entanglement in the parameter space – a behavior fundamentally distinct from the diagonal-only gauge structure of SaMA (Theorem D.4).

D.7.3 Pseudo code for SaMA implementation vs. MoRE implementation

As the permutations $\boldsymbol{P}_l$ and $\mathbf{P}_r$ are not explicitly constructed but we use matrix tranpose operation insead, we provide a pseudocode to demonstrate their implemetation in algorithm 2. We further provide pseudocode for MoRE [30] in algorithm 1 to help better distinguish the two methods.

Algorithm 1 MoRE Forward Pass

Require: Input $x \in \mathbb{R}^{B \times (b \times s)}$
Require: Factors $W_1 \in \mathbb{R}^{b \times r \times s}$, $W_2 \in \mathbb{R}^{b \times s \times r}$

Phase 1: Local Block Projection
▷ Reshape x to (B, b, s) and apply Batched MatMul
$y_1 \leftarrow \text{BLOCKDIAGMULT}(x, W_1)$ ▷ Shape: (B, b, r)

Phase 2: Middle Permutation (Stride Routing)
▷ Flatten, view as (r, b), then transpose
$y_p \leftarrow \text{REARRANGE}(y_1, \text{'}(r\ b) \rightarrow (b\ r)\text{'})$ ▷ Shape: (B, b, r)

Phase 3: Final Block Projection
$y_2 \leftarrow \text{BLOCKDIAGMULT}(y_p, W_2)$ ▷ Shape: (B, b, s)

Phase 4: Output Permutation
$y_o \leftarrow \text{REARRANGE}(y_2, \text{'}(b\ s) \rightarrow (s\ b)\text{'})$ ▷ Shape: (B, s, b)

return $\text{RESHAPE}(y_o, [B, s \times b])$

Algorithm 2 SaMA Forward Pass (Ours)

Require: Input $x \in \mathbb{R}^{B \times (c_L \times c_R)}$
Require: Factors $L \in \mathbb{R}^{r_R \times (r_L \times c_L)}$, $R \in \mathbb{R}^{c_L \times (r_R \times c_R)}$ (3D-tensors)
Require: Scaling factor α

Phase 1: Right Factor Projection
$y_r \leftarrow \text{BLOCKDIAGMULT}(x, R)$ ▷ Shape: (B, c_L, r_R)

Phase 2: Bilateral Permutation I (Mid-routing)
$y_{p1} \leftarrow \text{REARRANGE}(y_r, \text{'}(c_L\ r_R) \rightarrow (r_R\ c_L)\text{'})$ ▷ Shape: (B, r_R, c_L)

Phase 3: Left Factor Projection
$y_l \leftarrow \text{BLOCKDIAGMULT}(y_{p1}, L)$ ▷ Shape: (B, r_R, r_L)

Phase 4: Bilateral Permutation II (Missing in MoRE)
▷ **Restores spatial alignment via Kronecker commutation**
$y_{p2} \leftarrow \text{REARRANGE}(y_l, \text{'}(r_R\ r_L) \rightarrow (r_L\ r_R)\text{'})$ ▷ Shape: (B, r_L, r_R)

return $\text{RESHAPE}(y_{p2}, [B, r_L \times r_R]) \times \alpha$

Figure 13: Pseudo-code comparison detailing the tensor shape transformations. By explicitly tracking the dimensions, it becomes algebraically evident that SaMA incorporates a mandatory second bilateral permutation (Phase 4) to ensure proper output spatial alignment, a step entirely omitted in the MoRE architecture.

E What evidence supports disentangling the effects of SaMA adapter features

E.1 Expressivity via rank capacity

Here we discuss the role of rank capacity when considering 3 criteria:

- Low-rank regimes (when this trait is not dominant),
- SaMA's performance over ranks,
- SaMA's ability to overcome the bottleneck of high-rank regimes.

To isolate the rank effect of SaMA, we examine the model's performance on low-rank regions and take the regimes into account. In Fig. 19 (page 49), we demonstrate LoRA and SaMA's performance over varying parameter counts (associated with adapter ranks). It is clear that SaMA gains consistently increase as ranks increase. This basically suggests that their rank capacity contributes to the overall performance of adapters.

Meanwhile, the overall performance patterns of SaMA is higher than LoRA, especially in equal-rank comparison of low-rank regimes (for example $\text{LoRA}_{r=4}$ versus $\text{SaMA}_{2,2}$). It means that even in low-rank regimes, the recommended regions for LoRA, SaMA still performs well. This further suggests positive effects of SaMA which are beyond rank capacity. Here we tentatively postulate them as inductive biases of the Morpho structure as well as gauge symmetry.

Having said that, although rank capacity theoretically means a richer representation space of the adapter, the actual performance might not come with a straightforward translation into gains. The issue behind high-rank performance in dense matrices like LoRA is also illustrated in Fig. 19 (page 49) along with the previous studies from [29]. In these experiments, in high-rank regimes, LoRA either introduces a plateau region while leaving a significant gap to FFT or encounters a rank-collapse mode, making it suffer from poor behavior for complicated tasks.

Apart from that, given Fig. 23 (page 52), we can further analyze this phenomenon. Regarding effective ranks and singular value spectrum patterns, it is usually expected that the low-rank assumption underlying LoRA causes the adapter's singular values to concentrate in the most prominent directions. However, low-rank patterns in Fig. 23b-d, whose spectrum depletes quickly in high budgets, are not in alignment with FFT (as the reference).

In contrast, SaMA adapters tend to emulate the long-tailed spectrum patterns of FFT, while maintaining richer effective feature spaces. This consolidates the contribution of other mechanisms that support training dynamics rather than only static expressivity. Moreover, SaMA's performance in this case closely reaches the FFT adapter gains.

From these perspectives, it is reasonable to emphasize the contributions of expressivity as well as other remaining factors both directly and indirectly to the overall performance.

E.2 Morpho structure

Considering a naive diagonal matrix as a canonical example, this adapter (scaling transform only) theoretically possesses full rank and reduced parameter degeneracy space, but yields a hollow capacity as well as poor scalability.

To further make a comprehensive analysis, we compare SaMA with DiaBlo [18] (a standard single block diagonal matrix). Fundamentally, the distinction lies in the topology of information routing. An inherent limitation of block-sparse matrices is the isolation of far features, which does not facilitate global interaction in the high-dimensional representation spaces of LLMs (but small local groups instead). Table 11 (page 46) shows this effect on overall performance (Llama-13B and Llama2-7B).

Moreover, it is still a theoretical question of the reason for choosing the diagonal form for the block, rather than arbitrary arrangements (except for implementation convenience). They might introduce other paradigms of group-interacting. The Morpho structure specifically overcomes this blind spot by permutation, allowing global mixing. These empirical results advocate the contributions of the Morpho structure to the overall effectiveness.

Although general high-rank matrices do not absolutely mean improvement in expressivity, we further notice that: apart from intrinsic high-rank in diagonal matrices or DiaBlo adapter, which may lead to hollow high-ranks in extreme parameter budgets; SaMA takes this quantity into account by design,

making its rank grow as a quadratic function of the budgets. Thus we endorse this property due to its flexibility and rationale.

Table 7: Performance comparison of SaMA and DiaBlo on MetaMath tasks.

Method	#Params (M)	GSM8K	MATH
DiaBlo$_{N=128}$	46.4	72.18	28.92
DiaBlo$_{N=64}$	92.9	72.33	29.54
SaMA$_{16,16}$	54.0	78.45	39.45

To support this perspective, we further design experiments and compare performance of SaMA and DiaBlo in the MetaMath benchmark (a subset of 12K samples) as we consider the two constituent tasks as demanding NLP reasoning benchmarks that require rich representation capacity. All schemes were conducted on the same experimental platform. The results are reported in Table 7. Although these are not really comprehensive ablations, the prominent pattern here is that the overall performance degrades in sparse adapters regardless of parameter counts. Moreover, the gap is particularly considerable in the demanding tasks of MATH.

E.3 Invariant manifolds

Regarding the reduced gauge freedom, it serves as theoretically supportive evidence of an (Adam/AdamW) optimizer-friendly adapter design. In our toy experiment, Fig. 3 (Sec. 3.4, page 6) shows empirical support for the claim that the $GL(r)$ gauge freedom of LoRA creates an optimization bottleneck that Morpho's diagonal gauge structure inherently avoids. Furthermore, in Fig. 19, we conjecture that the smooth performance improvement towards FFT points might have contributions from the decoupling effect by reduced gauge freedom, which becomes severe when the budgets increase in LoRA. Without this, high-rank LoRA adapters are shown to not be able to manage the bottleneck as well as close the gap with FFT despite the fact that LoRA has dense components, allowing more freedom in internal interactions.

Table 8: Performance comparison of gauge-inflated and vanilla SaMA.

Method	#Params (M)	GSM8K	MATH
Gauge-Inflated SaMA	56.4	77.24	38.10
SaMA$_{16,16}$	54.0	78.45	39.45

To make things clear, we conduct additional comparison experiments where we artificially inflate gauge symmetry into the SaMA structure. In particular, rather than $\Delta\mathbf{W} = \boldsymbol{P}_l\mathbf{L}\boldsymbol{P}_r\mathbf{R}$, we introduce $(\mathbf{L} \leftarrow \mathbf{L}@\mathbf{L}', \mathbf{R} \leftarrow \mathbf{R}'@\mathbf{R})$, where $(\mathbf{L}', \mathbf{R}')$ are square block diagonal matrices of conformable shapes, which add degrees of freedom within each block components. The rank capacity and the SaMA structure are relatively preserved. The results are reported in Table 8.

This setting (SaMA $_{16,16}$) introduces a little overhead of 4.4% on parameter counts compared to the vanilla SaMA. The invariant space for each factor now is now a merely small batch of 16-by-16 matrices. All schemes are done on the same experimental platform. As we can see in this experiment results, the performance slightly degrades regardless of a marginal increase in terms of budgets.

F Further Experimental Result and Analysis

F.1 Matrix approximation – experimental settings

This section describes experiments in Sec. 3.2.

Baseline and setup In order to archieve a comprehensive understanding and fair evaluation, SaMA is compared to other reparameterization adapters including: FFT, the LoRA family [7, 28], QuanTA [11] with high-rank update, in a low-rank target matrix. We then simplify the target to a square matrix and set all its nonzero singular values to 1 to guarantee that it is low-rank but strictly of R, isolating the effect of spectral decay. In other words, R is not only the algebraic rank, but also the exact target effective rank [32].

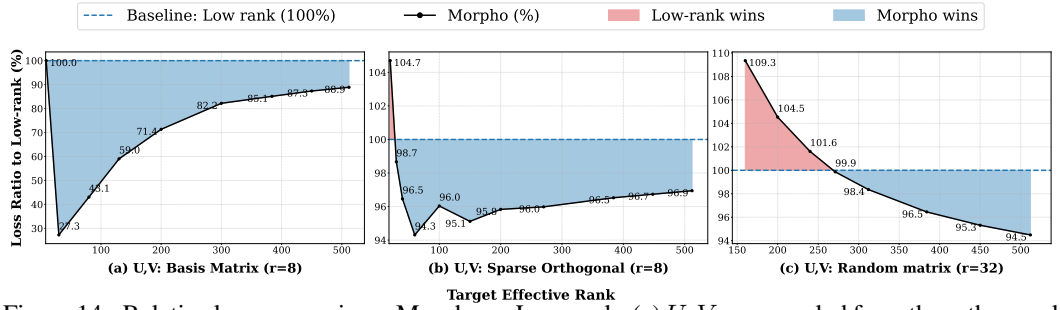

Figure 14: Relative loss comparison: Morpho vs Low-rank. (a) U, V are sampled from the orthogonal bases; (b) U, V are sampled from sparse orthogonal blocks; (c) U, V are sampled from random matrices. The crossover points R_{th} vary across settings. It is obvious that random matrix-based (U, V) is the worst case among all, requiring $R_{th} \approx 270$ in Fig. (c) $r = 32$; or $r = 217$ in Fig. 2$r = 8$.

In the experiments of matrix approximation, we use the AdamW optimizer[15] with the same learning rates of $1\mathrm{e}{-2}$, except for DoRA, with high learning rate of $4\mathrm{e}{-2}$ due to its slow convergence speed in such simulations, running 300 training iterations, then evaluate the average over 10 independent simulations with different random seeds. The main configurations are low-rank $r = 8$, and Morpho $(n_1, m_2) = (r, r)$, resulting in comparable parameter budgets in the compression problems. Moreover, we also compare low-rank final optimal points to SVD solutions by the Eckart–Young–Mirsky theorem [31] to make sure that the training processes lead to all points close to the theoretical approximation. The target matrices have dimensions of 1024×1024.

Entropy effective rank The standard algebraic rank of a matrix is highly sensitive to noise as infinitesimally small perturbations can easily lead to full apparent rank. In addition, it does not take into account and reflect the full singular value spectrum. To overcome these limitations, Roy and Vetterli [32] introduced an alternative continuous measurement of underlying information – effective rank (erank). Given a nonzero matrix $\mathbf{A}$ of shape $(M \times N)$, $K = \min M, N$, with all singular values $\{\sigma_i(\mathbf{A})\}$, the erank is calculated as follows:

$$p_i = \frac{\sigma_i}{\sum_{j=1}^{K} \sigma_j}, \quad \text{for } i = 1, 2, \ldots, K.$$

$$H(p) = -\sum_{i=1}^{K} p_i \ln(p_i) \qquad - \quad \text{Shannon Entropy,}$$

$$\text{erank}(\mathbf{A}) = r_{\text{eff}}(\mathbf{A}) = \exp[H(p)] = \exp\left(-\sum_{i=1}^{K} p_i \ln(p_i)\right). \tag{77}$$

The effective entropy rank provides a continuous capacity measure bounded by $1 \leq r_{\text{eff}} \leq K$. It reaches its minimum $r_{\text{eff}} = 1$ when the matrix is strictly rank-1, and its maximum $r_{\text{eff}} = K$ when all singular values are equal (an isotropic spectrum). In the context of machine learning, erank quantifies the true intrinsic capacity and information bottleneck of weight matrices, filtering out negligible noise directions that do not meaningfully contribute to the representation power. In our matrix approximation setting, the target matrix is already constructed to be algebraic low-rank; for that

reason, we tighten the design by imposing low-erank as a complementary metric without additional spectral collapse as well as rank deficiency.

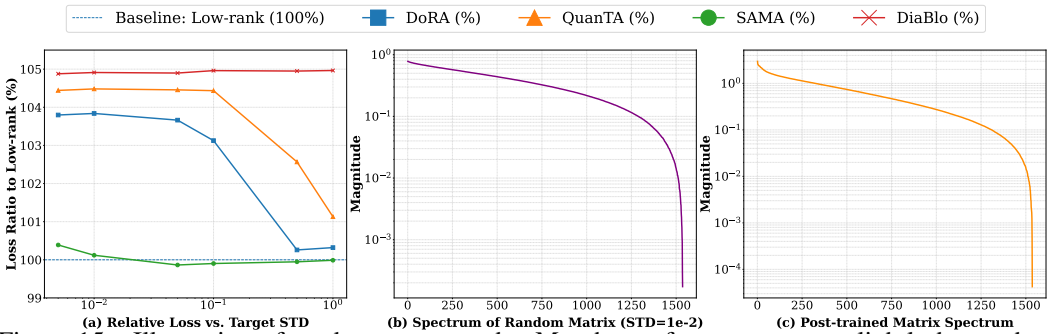

Figure 15: Illustration of random target mode. Morpho performs on par or slightly better than Low-rank approximation, and outperforms other baselines. (a) Comparison on expressivity of several adapters, given a random target. (b) Singular value spectrum of a matrix with random entries. (c) Singular value spectrum of a fine-tuned model weight. To make alignment with the reference matrix, and balance parameter budgets, we set dimension $d = 1536$, budgets equivalent to rank $r = 32$.

Target patterns and additional comparison results We also apply several settings of left and right orthogonal singular matrices (U, V) in order to control the geometric structure of the target updates. In the main experiment in Theorem 3.2 and Fig. 14 (c), (U, V) are independently sampled from random dense matrices, representing global and the least structured geometry. In addition, we create two special cases, where (U, V) are sampled from orthogonal bases, representing an extremely simple and highly structured case, shown in Fig. 14 (a) and block matrices of sparse orthogonal components, introducing local structure of overall orthogonality, shown in Fig. 14 (b). These three settings provide a spectrum from simple local geometry to fully isotropic geometry, allowing us to examine how the crossover point R_{th} varies with the structural complexity of the target matrix.

Moreover, we observe that random matrices share some basic characteristics with LLM's weights, being an important research interest in Deep learning. Hence, we adopt them as an exemplar target in our expressivity tests, then report the output in Fig. 15. It is obvious that given the comparable number of parameters, Low-rank approximation and Morpho parameterization considerably attain better compression than the other baselines. Moreover, Morpho parameterization in some cases slightly improves upon the Low-rank approximation, which is supported by the Eckart-Young theorem [31].

Next we will move to additional empirical results on downstream tasks.

F.2 Full results on the DROP dataset

In this section, we provide the full model-by-model comparison underlying the main-paper summary, with several experiments on large models. Table 9 provides detailed outputs, along with Fig. 4 in Sec. 4.1. Across all of our experiments, we train models for 2 epochs, except for the QuanTA experiments on Llama2-7B, which use 3 epochs to match the setting in the original paper [11].

Except for an outlier of $\text{SaMA}_{2,2}$, SaMA performance also steadily improves when the parameter budget increases and exhibits a nearly monotonic trend that suggests efficient parameter utilization. In these settings, to make a completely fair comparison between baselines, we only conduct full fine-tuning experiments on the corresponding modules, namely Query (`q_proj`) and Value (`v_proj`). As shown in the table, the gain with SaMA is clearly visible across low-to-high budget regimes of PEFT, where SaMA narrows or closes the gap to FFT more reliably than the other baselines.

F.3 Full results on the math reasoning task

Table 10 reports the complete arithmetic-reasoning results, extending the main-paper summary with full model-by-model and method-by-method comparisons. These detailed results complement the main discussion by showing the full ranking of baselines under aligned evaluation settings, and by making the relative strengths of lightweight and higher-capacity adapters more transparent.

Overall, the arithmetic-reasoning benchmark shows that SaMA maintains a favorable balance between accuracy and parameter efficiency across different settings. In particular, the detailed table confirms

Table 9: Benchmark of various fine-tuning methods on the DROP dataset. Baselines with † are from QuanTA [11]. Overall, SaMA consistently exhibits better performance compared to other baselines.

Model	FT Method	%Params	F_1 Score (↑)
Llama2_{7B}	FFT (`q_proj` and `v_proj`)	15.93	68.98
	Series†	0.747	58.8
	Parallel†	0.747	59.0
	$\text{LoRA}_{r=8}$	0.062	62.39
	$\text{LoRA}_{r=32}$	0.249	62.40
	$\text{LoRA}_{r=128}$	0.996	58.82
	$\text{QuanTA}_{16\text{-}8\text{-}8\text{-}4}$	0.041	62.09
	$\text{QuanTA}_{16\text{-}16\text{-}16}$	0.261	62.72
	$\textbf{SaMA}_{2,2}$	0.016	66.01
	$\textbf{SaMA}_{4,4}$	0.031	65.15
	$\textbf{SaMA}_{8,8}$	0.062	66.61
	$\textbf{SaMA}_{16,16}$	0.124	67.17
	$\textbf{SaMA}_{32,32}$	0.248	*67.95*
	$\textbf{SaMA}_{64,64}$	0.496	**68.07**
Llama2_{13B}	$\text{LoRA}^{\dagger}_{r=8}$	0.050	61.0
	$\text{QuanTA}^{\dagger}_{16\text{-}8\text{-}8\text{-}5}$	0.029	69.0
	$\textbf{SaMA}_{4,4}$	0.025	**73.12**

Table 10: Detailed arithmetic reasoning results across PEFT methods on Mistral-7B and Gemma-2 9B. This table extends the main-paper summary with full comparison along with parameter counts. Baselines (†) are from [35]. Best results are in **bold**, the runner-ups are in *italic*. In general, SaMA consistently surpasses other baseline regardless of lower budgets.

Method	Mistral-7B			Gemma-2 9B		
	# Params	GSM8K	MATH	# Params	GSM8K	MATH
Full FT†	7.24 B	63.87	17.65	9.24 B	79.23	38.02
$\text{LoRA}_{r=32}$	83.88 M	62.93	14.14	108.04 M	77.45	35.05
rsLoRA†	83.88 M	62.15	16.24	108.04 M	76.84	36.88
PiSSA†	83.88 M	62.43	16.52	108.04 M	77.12	37.04
DoRA†	85.26 M	62.65	16.64	109.88 M	77.58	37.04
LoRA-Pro†	83.88 M	63.07	17.32	108.04 M	78.26	37.53
HiRA†	83.88 M	63.15	17.44	108.04 M	78.47	38.22
$\text{ABBA}_{r=32}$	83.88 M	64.48	17.95	108.04 M	78.85	*39.59*
$\textbf{SaMA}_{8,8}$	10.49 M	67.70	*18.44*	13.51 M	*80.36*	39.36
$\textbf{SaMA}_{16,16}$	41.94 M	**68.23**	**19.18**	54.02 M	**81.58**	**39.76**

that the advantage of SaMA is not limited to a single configuration, but remains visible across multiple model scales and adapter budgets. This pattern is especially meaningful for arithmetic reasoning, where stronger performance typically requires both sufficient expressivity and stable optimization; the results therefore support our claim that SaMA provides a practical and robust alternative to conventional low-rank adapters in numerically demanding tasks.

F.4 A comprehensive benchmark of Commonsense reasoning

We further provide a comprehensive benchmark of the task in Table 11, including results of SaMA and a diverse set of PEFT methods across several LLaMA models. Figure 16 also demonstrates the main comparison across methods for each of these models, except for LLaMA-7B. Overall, it is apparent that SaMA outperforms most of the other baselines, having performance on par with FFT and sometimes surpassing it. Even in lightweight adapters, SaMA still performs well, with slight drops only, on these complex tasks, which illustrates its applicability in NLP problems.

Notably, the comprehensive benchmark in Table 11 highlights a clear and coherent trend: across model scales and variants, SaMA consistently delivers cutting-edge or tied-best average accuracy under substantially lower learnable-parameter budgets. SaMA achieves highly competitive and often the best performance across these subtasks of complicated reasoning without relying on FFT techniques like [81, 29]. The results also indicate that SaMA scales robustly and reliably from smaller to larger Llama variants, from the first generation to the current one of Llama3, preserving effectiveness rather

Table 11: Comprehensive benchmark of all FT methods on Commonsense reasoning, extending the main-paper summary with full comparisons. The best and runner-up results are highlighted in **bold**, and underline, respectively. Higher is better for all tasks. Symbols indicate where the results are taken from – $*$: [43], †: MiLoRA [64], $\perp$: KaSA [17], $\circ$: DoRA [28], $\bullet$: SMT [29], ♠: S^{2}FT [82], $\square$: HiRA [19], ▼: FlexLoRA [65] $\vee$: RepLoRA [10], ★: RefLoRA [72], ⅄: RandLoRA [51], ⊛: LaX [98], $\triangle$: DiaBlo [18], respectively. ▲: projected trainable parameters

Model	FT Method	#Params%	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	AVG(↑)
ChatGPT*	–	–	73.1	85.4	68.5	78.5	66.1	89.8	79.9	74.8	77.0
LLaMA-7B	FFT♠	100	70.3	84.2	80.1	92.3	85.4	86.6	72.8	83.4	81.9
	Prefix*	0.11	64.3	76.8	73.9	42.1	72.1	72.9	54.0	60.6	64.6
	Series*	0.99	63.0	79.2	76.3	67.9	75.7	74.5	57.1	72.4	70.8
	Parallel*	3.54	67.9	76.4	78.8	69.8	78.9	73.7	57.3	75.2	72.3
	LoRA♠	0.83	69.2	81.7	78.4	83.4	80.8	79.0	62.4	78.4	76.7
	RefLoRA★	0.83	69.6	82.5	79.5	88.3	82.6	81.6	66.6	80.2	78.9
	RepLoRA$^\vee$	1.01	71.8	84.1	79.3	85.2	83.3	82.4	66.2	81.2	79.1
	DoRA$^\circ_{r=16}$	0.43	70.0	82.6	79.7	83.2	80.6	80.6	65.4	77.6	77.5
	DoRA$^\circ_{r=32}$	0.84	69.7	83.4	78.6	87.2	81.0	81.9	66.2	79.2	78.4
	Galore♠	0.83▲	68.6	79.0	78.5	84.7	80.1	80.3	62.1	77.3	76.3
	LaX$^⊛_{32}$	0.83	69.6	81.9	78.9	84.7	80.8	79.8	64.8	78.0	77.3
	LISA♠	9.91	70.4	82.1	78.7	92.4	82.9	84.9	70.2	78.4	80.0
	S^{2}FT♠	0.81	72.7	83.7	79.6	93.4	83.5	86.1	72.2	83.4	81.8
	SMT$^\bullet$	0.84	68.7	81.7	78.3	91.6	78.8	84.1	68.7	77.4	78.7
	SMT$^\bullet$*(Best)*	4.91	72.0	82.9	80.7	93.3	82.4	86.1	70.6	83.0	81.4
	SaMA$_{4\times4}$	0.104	69.9	84.4	81.3	94.0	83.5	86.8	71.2	84.9	82.0
	SaMA$_{16\times16}$	0.416	71.1	84.9	81.7	94.1	84.0	87.6	72.7	86.2	**82.8**
LLaMA-13B	FFT♠	100	74.5	86.3	81.3	94.4	86.9	89.7	77.9	88.8	85.0
	Prefix*	0.03	65.3	75.4	72.1	55.2	68.6	79.5	62.9	68.0	68.4
	Series*	0.80	71.8	83.0	79.2	88.1	82.4	82.5	67.3	81.8	79.5
	Parallel*	2.89	72.5	84.8	79.8	92.1	84.7	84.2	71.2	82.4	81.5
	LoRA*	0.67	72.1	83.5	80.5	90.5	83.7	82.8	68.3	82.4	80.5
	RepLoRA$^\vee$	0.99	73.1	85.2	84.7	91.1	85.9	84.7	73.4	85.6	82.9
	DoRA$^\circ_{r=16}$	0.35	72.5	85.3	79.9	90.1	82.9	82.7	69.7	83.6	80.8
	DoRA$^\circ_{r=32}$	0.68	72.4	84.9	81.5	92.4	84.2	84.2	69.6	82.8	81.5
	LaX$^⊛_{32}$	0.67	71.3	85.4	81.3	91.3	84.1	84.4	71.8	83.1	81.6
	S^{2}FT♠	0.65	74.2	85.7	80.7	94.9	86.4	88.4	76.3	87.8	84.3
	SMT$^\bullet$	0.68	71.1	84.4	81.7	93.7	83.2	86.7	73.7	85.2	82.4
	SMT$^\bullet$*(Best)*	4.91	72.6	86.1	81.9	95.0	86.1	88.2	77.1	87.4	84.3
	DiaBlo$^\triangle_{N=64}$	1.06	74.7	86.9	81.2	91.5	87.4	89.8	78.2	90.2	84.9
	SaMA$_{4\times4}$	0.084	72.9	87.4	83.6	95.5	87.2	90.3	78.2	88.7	**85.5**
	SaMA$_{16\times16}$	0.336	73.5	87.6	82.6	95.4	87.7	90.2	78.0	88.3	85.4
LLaMA2-7B	FFT♠	100	74.7	84.9	78.7	93.7	84.1	87.5	75.2	85.0	83.0
	LoRA$^\circ$	0.83	69.8	79.9	79.5	83.6	82.6	79.8	64.7	81.0	77.6
	RefLoRA★	0.83	72.5	83.8	80.0	87.0	84.9	86.4	71.5	80.2	80.8
	PiSSA†	0.83	67.6	78.1	78.4	76.6	78.0	75.8	60.2	75.6	73.8
	MiLoRA†	0.83	67.6	83.8	80.1	88.2	82.0	82.8	68.8	80.6	79.2
	DoRA$^\circ_{r=16}$	0.42	72.0	83.1	79.9	89.1	83.0	84.5	71.0	81.2	80.5
	DoRA$^\circ_{r=32}$	0.84	71.8	83.7	76.0	89.1	82.6	83.7	68.2	82.4	79.7
	HiRA$^\square$	0.83	71.2	83.4	79.5	88.1	84.0	86.7	73.8	84.6	81.4
	KaSA$^\perp$	0.84	73.6	84.4	80.2	91.5	84.5	84.7	72.1	81.2	81.5
	S^{2}FT♠	0.81	72.9	86.1	80.2	94.3	85.5	87.2	74.6	83.4	83.0
	SMT$^\bullet$	0.84	72.0	83.8	80.8	93.3	82.8	86.7	74.0	81.0	81.8
	SMT$^\bullet$*(Best)*	4.91	72.6	85.2	82.0	94.4	85.7	87.8	74.5	85.0	83.4
	DiaBlo$^\triangle_{N=64}$	1.04	73.9	84.8	81.7	90.0	85.0	87.9	76.8	86.8	83.4
	SaMA$_{4\times4}$	0.104	73.1	85.5	81.8	94.7	85.0	88.7	73.9	86.0	83.6
	SaMA$_{16\times16}$	0.42	73.9	86.2	81.9	94.9	86.4	89.1	77.0	86.8	**84.5**
LLaMA3-8B	FFT♠	100	73.9	86.2	79.1	93.1	85.8	88.1	78.2	84.0	83.6
	LoRA$^\circ$	0.70	70.8	85.2	79.9	91.7	84.3	84.2	71.2	79.0	80.8
	RefLoRA★	0.70	75.5	89.7	81.1	95.6	87.3	91.0	79.8	86.0	85.8
	PiSSA†	0.70	67.1	81.1	77.2	83.6	78.9	77.7	63.2	74.6	75.4
	MiLoRA†	0.70	68.8	86.7	77.2	92.9	**85.6**	86.8	75.5	81.8	81.9
	DoRA$^\circ_{r=16}$	0.35	74.5	88.8	80.3	95.5	84.7	90.1	79.1	87.2	85.0
	DoRA$^\circ_{r=32}$	0.71	74.6	89.3	79.9	95.5	85.6	90.5	80.4	85.8	85.2
	HiRA$^\square$	0.70	75.4	89.7	81.2	95.4	87.7	93.3	82.9	88.3	86.7
	FlexLoRA▼	0.70	72.8	89.1	80.7	96.0	86.4	90.8	81.3	87.2	85.5
	KaSA$^\perp$	0.71	73.6	88.1	80.4	94.7	85.5	89.7	79.4	85.6	84.6
	S^{2}FT♠	0.70	75.0	89.0	80.7	96.5	88.0	92.5	83.4	87.8	86.6
	RandLoRA$^⅄_{30}$	0.7	76.3	88.1	80.3	95.7	86.1	90.4	80.9	87.0	85.6
	RandLoRA$^⅄_{15}$	1.4	72.6	88.0	81.4	95.7	87.8	91.3	80.9	89.0	85.8
	SMT$^\bullet$	0.71	75.7	88.4	81.4	96.2	88.2	92.7	83.2	88.6	86.8
	SMT$^\bullet$*(Best)*	3.01	75.1	89.9	82.4	96.3	88.8	92.6	82.8	89.6	87.2
	SaMA$_{4\times4}$	0.088	73.9	90.6	82.8	96.6	88.2	93.7	83.6	89.6	87.4
	SaMA$_{16\times16}$	0.351	75.4	90.3	83.2	96.7	88.7	94.1	83.8	90.1	**87.8**

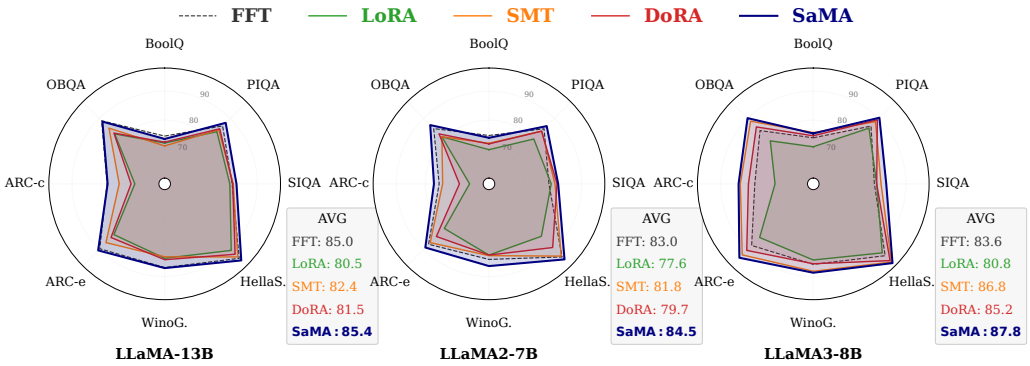

Figure 16: Task-wise performance comparison across **FFT(black)**, **LoRA(green)**, **SMT(orange)**, **DoRA(red)** and **SaMA (blue)** on the Llama family. SaMA only uses half as many parameters in these experiments, but maintains more accurate outputs across most of the tasks.

than showing any signs of trade-off in lightweight adapters. Furthermore, SaMA also demonstrates the effectiveness and robustness over higher ranks, without any degradation due to optimization. Moreover, the benchmark indicates that SaMA offers a favorable balance of expressivity, omni-rank, parameter-efficiency, and practical scalability for modern PEFT.

For example, on Llama-13B, $\text{SaMA}_{16,16}$ reaches an average score of 85.4 with only 0.336% parameters, compared with LoRA at 80.55 with 0.67%, or FFT at 85.0. On Llama2-7B, $\text{SaMA}_{16,16}$ attains 84.5 on average with 0.42% parameters, compared with SMT at 81.8 under 0.84% (and 83.4 at SMT-Best with 4.91%), or FFT at 83.0. On Llama3-8B, $\text{SaMA}_{16,16}$ reports 87.8 average with 0.351% parameters, or even $\text{SaMA}_{4,4}$ achieves 87.4 with only 0.09% parameters, while $\text{DoRA}_{r=32}$ and HiRA with about 0.71% report 85.2 and 86.7, respectively.

F.5 Attention fine-tuning on Commonsense reasoning

Table 12: Comparison of SaMA and QuanTA in Q-V attention fine-tuning settings. Results with a symbol $\star$ are from QuanTA paper [11]. Across all model scales, although the configurations are set with comparable or 20-40% lower parameter budgets, SaMA stably results in superior performance. This demonstrate the substantial generalization capability of our method.

Model	Method	#Params%	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	AVG
ChatGPT	–	–	73.1	85.4	68.5	78.5	66.1	89.8	79.9	74.8	77.0
LLaMA-7B	QuanTA*	0.041	71.6	83.0	79.7	91.8	81.8	84.0	68.3	82.1	80.3
	SaMA$_{4\times4}$	0.031	70.8	84.0	81.4	93.8	83.4	86.7	71.3	84.0	**81.9**
LLaMA-13B	QuanTA*	0.029	73.2	85.4	82.1	93.4	85.1	87.8	73.3	84.4	83.1
	SaMA$_{4\times4}$	0.025	72.6	87.4	83.3	95.6	85.6	88.4	76.2	88.4	**84.7**
LLaMA2-7B	QuanTA*	0.041	72.4	83.8	79.7	92.5	83.9	85.3	72.5	82.6	81.6
	SaMA$_{4\times4}$	0.031	72.5	85.1	81.2	94.4	84.8	87.9	75.0	84.5	**83.2**
LLaMA2-13B	QuanTA*	0.041	75.8	86.9	81.2	94.4	87.0	89.6	77.9	85.2	84.8
	SaMA$_{4\times4}$	0.025	75.4	87.3	82.9	96.0	88.0	91.5	80.8	86.2	**86.0**
LLaMA3-8B	QuanTA*	0.035	74.3	88.1	81.8	95.1	87.3	91.1	81.7	87.2	85.8
	SaMA$_{4\times4}$	0.021	74.0	90.1	82.1	96.2	87.2	93.2	81.7	87.0	**86.4**

Looking closely into details of QuanTA experiments [11], it is noticeable that they added adapters into Q-V modules only in each transformer layer in order to keep the implementation of tensor factorization for square matrices simple. Given that both methods are adapter-oriented, we argue that it is much fairer to adapt them to the same modules while fine-tuning foundation models. Hence, we conduct auxiliary experiments on these Llama model families (including Llama2-13B). That is to establish an equitable assessment between them. The outputs are then reported in Table 12.

F.6 Efficiency validation

To make a comparison of adapter efficiency, we align the configurations of all methods to match the number of trainable parameters. The tests are conducted across several foundation models of different sizes so that they produce an unbiased judgment. The reports are constructed by monitoring

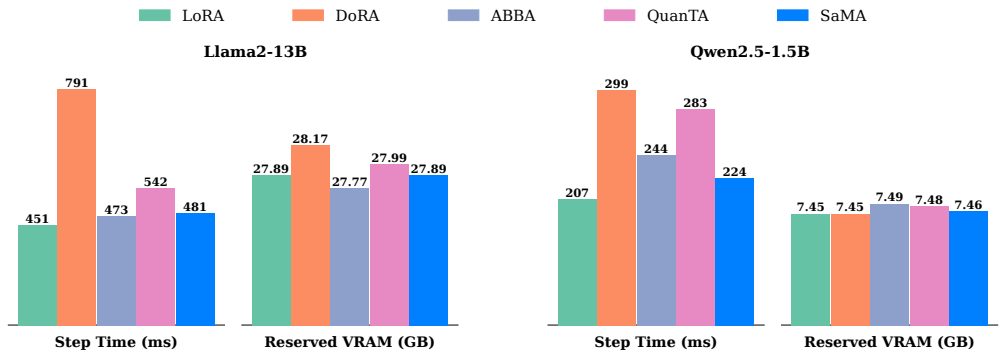

Figure 17: Time complexity and memory footprint on Qwen2.5-1.5B and Llama2-13B. SaMA generally maintains a memory footprint nearly identical to LoRA, while incurring only a modest step-time overhead, at less than 9 per cent, and remains substantially faster than DoRA and QuanTA.

time per iteration and reserved memory after 50 warmup steps, averaged 3 times over 25 iterations, while other settings are kept the same across all the experiments.

In particular, we consider a baseline set including LoRA [7], QuanTA [11], SaMA, as 3 distinct representatives of fine-tuning via reparameterization. In addition, we adopt ABBA [35] – Hadamard product of low-rank matrices, and DoRA [28] – an improvement of LoRA due to the prevalence of the LoRA family.

Besides Fig. 5, additional results are also reported in Fig. 17. Similar to the pattern in Fig. 5, there exists consistency in the relative resource profiles of these approaches in regard to computational time and memory consumption across different model scales. These results indicate that the structured higher-rank capacity of SaMA introduces little additional computational and memory overhead in practice.

F.7 Ablation study on adapter initialization

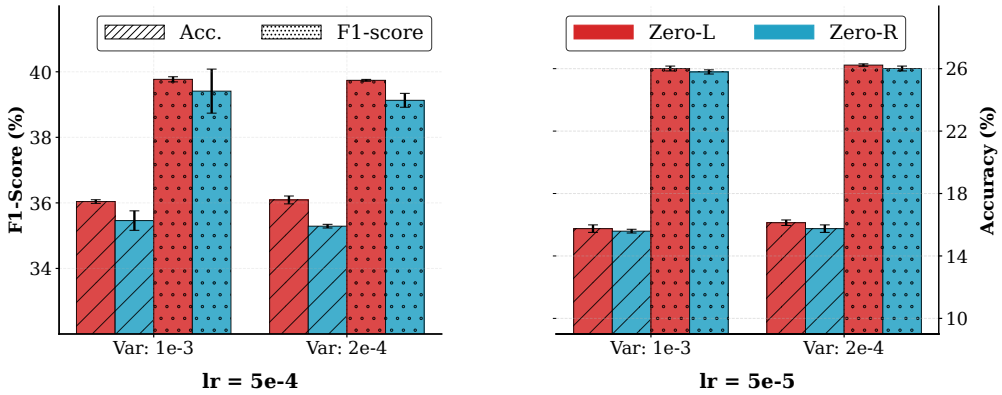

Figure 18: Sensitivity analysis on initialization schemes of SaMA adapters. Either of the two factors is equal to zero, while the remaining factor follows a normal distribution with variance on the horizontal axes. We use both metrics: accuracy (right axis) and F1-score (left axis) for comparison.

For these experiments, we fine-tune Qwen2.5-1.5B with a fixed SaMA configuration of $(4, 4)$ for three attention modules: `q_proj, k_proj, v_proj`. The training set is a 600-sample subset of DROP. The final checkpoints are evaluated after 1 epoch. To improve the robustness of the final conclusion, we consider multiple learning rates and initialization variances. Table 3 in Sec. 4.6 and Fig. 18 provide the quantitative outputs and visualization, respectively.

Observation: Across these settings, SaMA with zero-$\mathbf{L}$ initialization consistently yields better accuracy and F1-scores compared to zero-$\mathbf{R}$, at about $1-2\%$. Furthermore, zero-$\mathbf{L}$ demonstrates superior stability; it achieves a remarkably lower variance across most experiments, with a small standard deviation (std) of 0.25.

The results also advocate another critical insight regarding robustness. While performance margins exist between different configurations, our approach exhibits a remarkable intrinsic robustness and stabilizing effects, maintaining reliable efficacy regardless of initialization schemes.

F.8 SaMA performance when increasing parameter budgets

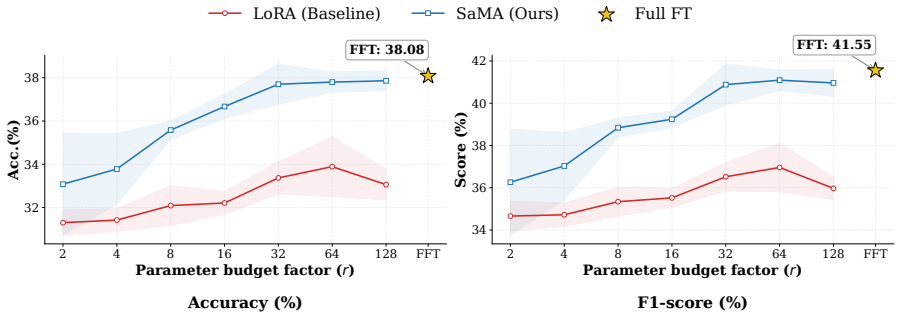

Figure 19: Performance comparison between SaMA and LoRA; FFT (on Query and Value modules) serves as the upper-bound baseline. The plots show Accuracy (%) and F1-score (%) across various parameter budget factor r. Shaded regions represent standard deviations.

We further evaluate the performance of SaMA against LoRA and FFT using the Qwen2.5-1.5B model [6] on the DROP dataset. The primary goal is to analyze how performance scales under different parameter budgets (r) – equivalent to LoRA rank r and $(n_1, m_2) = (r, r)$ in SaMA. In all settings, we only train Query and Value modules to make a completely fair comparison.

The key finding is that SaMA performs better when the parameter counts increase, and steadily approaches FFT results. Meanwhile, LoRA, in such case, exhibits a performance gap to FFT, regardless of adding more budget. A similar observation is also reported in [29], suggesting more comprehensive experiments to investigate in future work.

F.9 Fine-tuning Vision transformer

Along with the main benchmark on complicated NLP downstream tasks on large foundation models, we further demonstrate the versatility and effectiveness of SaMA in vision domains. Particularly, we extend our evaluation in image classification by post-training the ViT-B/16 architecture [2], pretrained on the ImageNet-21K [99] on VTAB-1K [100]. Although the classification task and baseline model are not sophisticated, we still gain overall judgment of SaMA in lightweight modes for common problems. The VTAB-1K [100] benchmark is a diverse suite containing 19 datasets. There are 3

Table 13: Performance comparison of SaMA with the baseline and efficient methods on the VTAB-1k classification benchmark with ViT-B/16 86M pre-trained on ImageNet-21k. The best results among these baselines are in **bold**, excluding SSF* which utilizes augmented ViT backbone by AugReg. All baseline results are from [101].

Methods	Natural								Specialized					Structured											
	CIFAR-100	Caltech101	DTD	Flowers102	Pets	SVHN	Sun397	**Mean**	Camelyon	EuroSAT	Resisc45	Retinopathy	**Mean**	Clevr-Count	Clevr-Dist	DMLab	KITTI-Dist	dSpr-Loc	dSpr-Ori	sNORB-Azim	sNORB-Ele	**Mean**	**Mean Total**	**Params.(M)**
Full fine-tuning	68.9	87.7	64.3	97.2	86.9	87.4	38.8	75.9	79.7	95.7	84.2	73.9	83.4	56.3	58.6	41.7	65.5	57.5	46.7	25.7	29.1	47.6	65.6	85.80
Linear probing	63.4	85.0	63.2	97.0	86.3	36.6	51.0	68.9	78.5	87.5	68.6	74.0	77.2	34.3	30.6	33.2	55.4	12.5	20.0	9.6	19.2	26.9	52.9	0.04
Adapter	74.1	86.1	63.2	97.7	87.0	34.6	50.8	70.5	76.3	88.0	73.1	70.5	77.0	45.7	37.4	31.2	53.2	30.3	25.4	13.8	22.1	32.4	55.8	0.27
Bias	72.8	87.0	59.2	97.5	85.3	59.9	51.4	73.3	78.7	91.6	72.9	69.8	78.3	61.5	55.6	32.4	55.9	66.6	40.0	15.7	25.1	44.1	62.1	0.14
VPT-Shallow	77.7	86.9	62.6	97.5	87.3	74.5	51.2	76.8	78.2	92.0	75.6	72.9	79.7	50.5	58.6	40.5	67.1	68.7	36.1	20.2	34.1	47.0	64.9	0.11
VPT-Deep	**78.8**	90.8	65.8	98.0	88.3	78.1	49.6	78.5	81.8	96.1	83.4	68.4	82.4	68.5	60.0	46.5	72.8	73.6	47.9	32.9	37.8	55.0	69.4	0.60
LoRA	67.1	91.4	69.4	98.8	90.4	85.3	54.0	79.5	84.9	95.3	84.4	73.6	84.6	82.9	**69.2**	49.8	78.5	75.7	47.1	31.0	**44.0**	59.8	72.3	0.29
AdaptFormer	70.8	91.2	70.5	99.1	90.9	86.6	54.8	80.6	83.0	95.8	84.4	76.3	84.9	81.9	64.3	49.3	80.3	76.3	45.7	31.7	41.1	58.8	72.3	0.16
FacT-TK$_{<32}$	70.6	90.6	70.8	99.1	90.7	88.6	54.1	80.6	84.8	**96.2**	84.5	75.7	85.3	82.6	68.2	49.8	80.7	80.8	47.4	33.2	43.0	60.7	73.2	0.07
ARC	72.2	90.1	72.7	99.0	91.0	**91.9**	54.4	81.6	84.9	95.7	**86.7**	75.8	85.8	80.7	67.1	48.7	81.6	79.2	51.0	31.4	39.9	60.0	73.4	0.13
RLRR	75.6	**92.4**	**72.9**	**99.3**	91.5	89.8	57.0	***82.7***	86.8	95.2	85.3	75.9	85.8	79.7	64.2	**53.9**	**82.1**	83.9	53.7	**33.4**	43.6	**61.8**	74.5	0.33
SSF*	69.0	92.6	75.1	99.4	91.8	90.2	52.9	81.6	87.4	95.9	87.4	75.5	86.6	75.9	62.3	53.3	80.6	77.3	54.9	29.5	37.9	59.0	73.1	0.24
SaMA$_{2-2}$	75.3	**92.4**	72.53	**99.2**	**92.0**	89.4	**57.8**	***82.7***	**88.2**	95.8	86.5	**76.4**	***86.7***	**83.0**	63.4	53.0	81.4	**84.7**	**53.8**	33.0	41.4	61.7	**74.7**	0.33

main groups: Natural, Specialized, and Structured. While these datasets cover various potential facets such as classification, object categories, distinct semantics; each task includes only 1000 images for training.

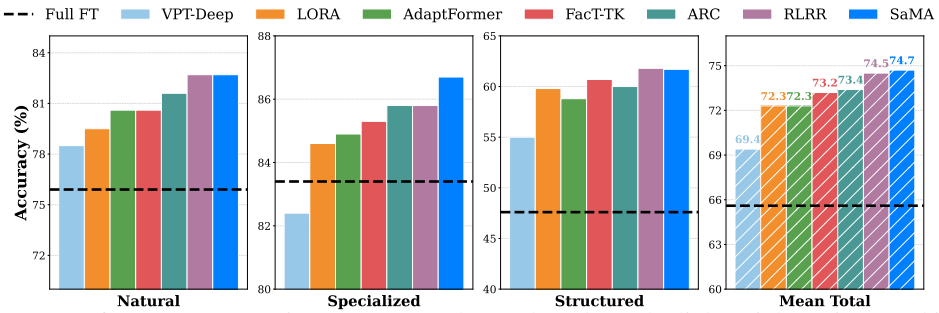

Figure 20: Performance Comparison on VTAB-1k Benchmark. In the lightweight regimes and image classification tasks, SaMA uniformly demonstrates effectiveness in terms of group performance and global results.

The final outcomes are detailed in Table 13, highlighting the superior performance of SaMA across most settings and in overall metrics. On average, SaMA with lightweight adapters yield results that are competitive or on par with the baselines, with notable gaps – exceeding the others on the final scores. We further illustrate the performance on each group and the total sets of noticeable methods in Fig. 20 where SaMA performance trends are consistent in each group.

G Ablation studies

To afford deeper insight into fine-tuning with Morpho matrices, we examine extensive ablation studies on SaMA and carry out pertinent analyses, further elucidating critical aspects of the method.

G.1 Module-specific adaptation

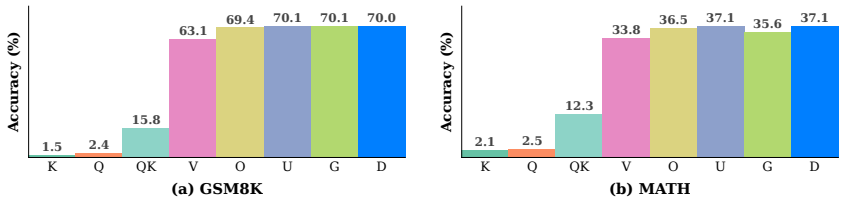

Figure 21: Comparative contribution of individual components in transformer layers: Key, Query, Value, Output, Up, Gate and Down projections to SaMA fine-tuning performance.

Figure 21 investigates the impact of fine-tuning a single module in transformer layers of Qwen2.5-1.5B with SaMA adapters, while keeping the number of trainable parameters fixed across settings.[5] A clear pattern emerges across both tasks: tuning only Query or Key contributes very little, whereas adapting the Value and Output projections results in substantially larger gains. One possible explanation is that adapting Query or Key only changes attention weights, but does not directly update the value content being propagated, which may limit their effectiveness. Further rigorous research might be needed to provide a comprehensive analysis.

Meanwhile, the strongest results are consistently obtained in the `MLP` modules, indicating the most impactful factors. For example, on GSM8K, they all reach around 70%, compared to 63% – 69% in `v_proj` and `o_proj`, respectively. The trend is similar on MATH – a more complex task, where Up and Down-tuning reach 37%. These reflect the most functional roles of feed-forward layers: neural memory [102]. They store and adapt task-relevant procedural knowledge in math. However, the gaps between `up_proj` and `down_proj` are small, suggesting an overall balance between their functions.

G.2 Asymmetric configuration of (n_1, m_2)

In Sec. 2, each SaMA adapter is configured by two primary hyperparameters (n_1, m_2), whose interaction controls both the intermediate rank and the total trainable parameters. An important question is how to choose the two factors from the final rank $r = n_1 m_2$. Since it is recommended to use (n_1, m_2) as divisors of input/output dimensions – to avoid unnecessary padding, it is more straightforward to compare different configurations that result in the same rank, as reported in Table 14.

Table 14: SaMA performance with different configurations of hidden rank 64.

$(n_1 - m_2)$	**#Params**	F_1**-score**$_{\pm\text{std}}$
$(16 - 4)$	3.28M	$32.61_{\pm 1.03}$
$(8 - 8)$	2.62M	$32.58_{\pm 0.60}$
$(4 - 16)$	3.28M	$31.9_{\pm 1.56}$

Overall, with the same training schemes, the symmetric configuration $(n_1 = m_2)$ achieves performance on par with the best asymmetric settings, while providing a more favorable trade-off between parameter efficiency and capability. Moreover, it also shows significant improvement in robustness across random seeds. In the `MLP` layers here, the asymmetric configurations require about 25% more parameters than $(8, 8)$, yet do not yield consistent gains and exhibit noticeably higher variance across random initializations. By contrast, the symmetric setting offers a better balance between efficacy, parameter cost, and stability, making it the more practical default choice in most experiments.

H Understanding the adapter via mechanistic analysis

H.1 Does SaMA leverage the Omni-rank property?

To gain deeper insight beyond the main benchmark scores, we further carry out post-hoc analyses on adapters from two complementary perspectives: structure of learned representation and optimization dynamics by fine-tuning Qwen2-1.5B on a subset of DROP.

[5] Except for **Query-Key** (QK) fine-tuning, which is 10% more than the others.

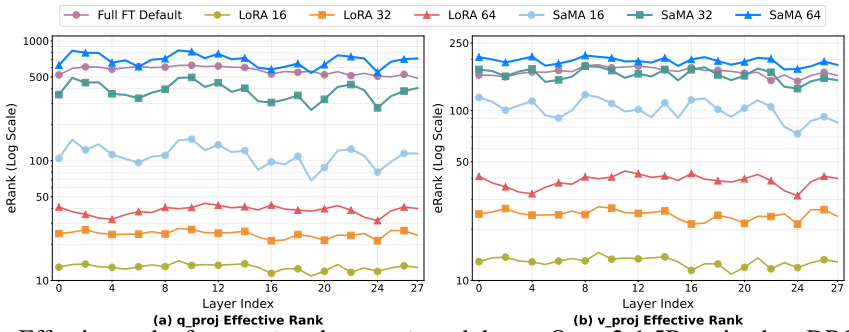

Figure 22: Effective rank of `q_proj` and `v_proj` modules on Qwen2-1.5B, trained on DROP. SaMA utilizes higher nominal capacity and converts it into a set of rich update features.

While Theorem 3.1 provides a generic formula for the adapter nominal ranks, a natural question is whether the higher rank is translated into the learned update, or whether the optimization collapses the solution into a much lower-dimensional subspace when the parameter budget increases. In such cases, the fine-tuning adapter is really in low dimensions. To examine this, we analyze the effective rank [32] (See Appendix F.1) of the adapter weights and therefore provides a sharp view of how hidden features span in practice.

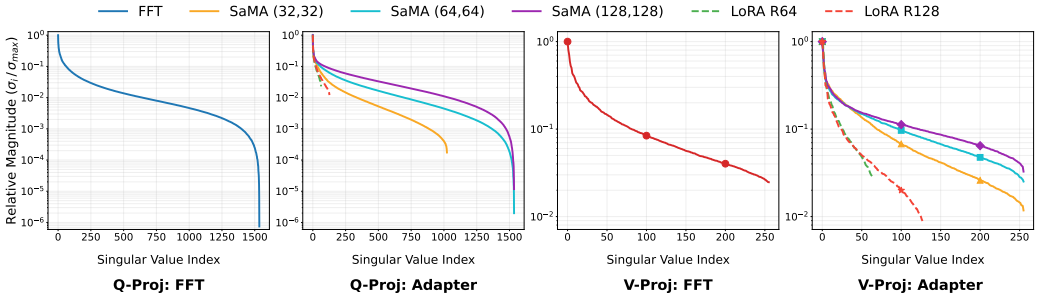

Figure 23: Singular value spectrum comparison between SaMA and FFT on Q,V modules. Complementing Fig. 22, SaMA shares richer long-tailed spectrum patterns in common with FFT, being able to achieve a diverse singular value profile while LoRA shows a sharper energy concentration

As shown in Fig. 22, SaMA tends to preserve intense long-tailed spectrum records in both `q_proj` and `v_proj` modules, and might surpass the effective rank of FFT adapter[6] if there are sufficient parameters. It indicates that: (i) the learned updates occupy a broader subspace rather than concentrate most into a few dominant directions. (ii) the long-tailed singular values really matter in matrix expressiveness, hence increasing adapter rank is a means of making performance improvement. This observation is consistent with the point that SaMA better exploits its allocated capacity and is less prone to dimensional collapse due to the "strictly low intrinsic rank assumption". This interpretation is further supported by the singular value spectra in Fig. 23. Apart from low-rank updates, SaMA exhibits a less concentrated but long-tailed spectrum by virtue of higher rank property, while remaining qualitatively close to FFT. These paradigm suggests that the nominal capacity of SaMA is not merely theoretical, but translated into a rich set of learned features in practice. In addition, compared to SaMA and FFT, LoRA exhibits a faster spectral decay, even at relatively higher ranks such as $r = 64$ or 128.

H.2 Gradient behavior during optimization

As shown in Fig. 24, the loss curves of different adapters remain mainly comparable, whereas their gradient behavior is more clearly separated. In particular, SaMA exhibits more stable gradient norms, indicate flat minima [103]. The key distinction therefore lies not only in the final loss, but in the landscape as well as the optimization path taken to reach them that have critical effects on generalizability.

[6] We subtract the pre-trained matrices from the fine-tuned matrices, then analyze the adapter $\Delta\mathbf{W}$

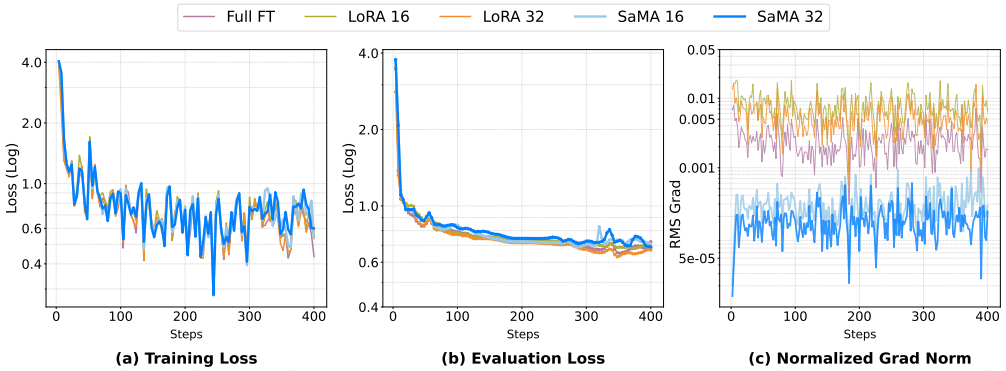

Figure 24: Fine-tuning metric comparison: (a) Training loss. (b) Evaluation loss (c) Normalized Gradient norm on DROP. Although different adapters achieve broadly similar loss trajectories, SaMA appears more stable in gradient behavior, consistent with better-conditioned optimization dynamics.

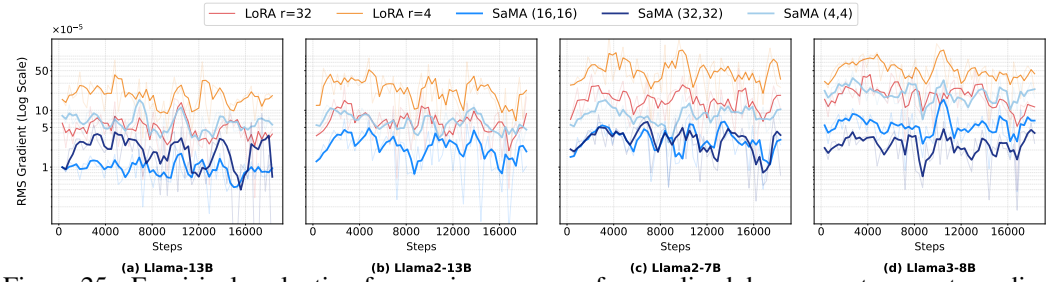

Figure 25: Empirical evaluation for moving average of normalized, by parameter count, gradient norms of full commonsense training (raw gradient data in transparent lines). SaMA exhibits lower normalized gradient norms. The more parameter counts, the more norms tend to decrease, suggesting flatter minima and stabilizing effects.

We further investigate this phenomenon on the full commonsense reasoning experiments across Llama family, illustrating the output in Fig. 25. The lower normalized gradient norms of SaMA are broadly in consistent with better-conditioned optimization dynamics under a less degenerate parameterization [104]. This observation also aligns with a long line of work relating gradients and flatter solutions to improve generalization gaps [105] via flat minima [103], as well as with optimization methods that explicitly favor flatter neighborhoods [106]. However, there are many facets that need to be thoroughly examined in the future work, since the complicated relationship between flatness and generalization also depends on the parameterization and the notion of sharpness [104] used while different adapter designs induce distinct invariant manifolds and degrees of gauge symmetry (See Sec. 3.4 and Appendix D.6).

I Dataset and experimental details

I.1 Details of datasets on experiments

Dataset	Task	# Train	# Val	# Test	Metric	Answer
DROP [37]	Reading comprehension with discrete reasoning	2000	800	1200	F_1-Score	Phrase
COMMONSENSE147K	Commonsense reasoning (Train)	146627	1000	–	–	–
BoolQ	Commonsense reasoning (Test)	–	–	3270	Accuracy	Yes/No
PIQA	Commonsense reasoning (Test)	–	–	1838	Accuracy	Option
SIQA	Commonsense reasoning (Test)	–	–	1954	Accuracy	Option
HellaSwag	Commonsense reasoning (Test)	–	–	10042	Accuracy	Option
WinoGrande	Commonsense reasoning (Test)	–	–	1267	Accuracy	Option
ARC-Easy	Commonsense reasoning (Test)	–	–	2376	Accuracy	Option
ARC-Challenge	Commonsense reasoning (Test)	–	–	1172	Accuracy	Option
OBQA	Commonsense reasoning (Test)	–	–	500	Accuracy	Option
METAMATH (20K)	Arithmetic reasoning (Train)	20000	–	–	–	–
GSM8K	Arithmetic reasoning (Test)	–	–	1319	Accuracy	Number
MATH	Arithmetic reasoning (Test)	–	–	5000	Accuracy	Number

Table 15: Statistics of Datasets for Reading Comprehension, Commonsense, and Arithmetic Reasoning.

DROP [37] is a reading comprehension benchmark designed to test discrete reasoning over natural language passages. Unlike traditional Q&A datasets, DROP requires models to perform operations such as counting, addition, and comparison to answer correctly. In [11], the dataset is used to evaluate the performance of high-rank update with QuanTA.

Following the QuanTA experimental setup [11], we take 2800 samples from the training set, and 1200 samples from the validation set as our test set since there is no official public test set on Hugging Face. Then, the 2800-sample raw training set is partitioned into a training set with 2000 samples, and a validation set with 800 samples. Moreover, F_1 score is also used as a metric for this task, which measures the closeness of the generative output compared to the ground truth answer.

Commonsense-147k is a comprehensive training collection that is refined from commonsense170k [43] by removing all duplicates. It contains a mixture of data samples from 8 different benchmark datasets in the domain:

- **BoolQ** [107] dataset: consists of naturally occurring yes/no questions, generated in unprompted and unconstrained settings. The context passage in the original triple is eliminated.
- Physical Interaction: Question Answering or **PIQA** [108] dataset: comprises samples structured as multiple-choice questions, each presenting a question with two possible solutions that require physical commonsense to answer.
- Social-IQA or **SIQA** [109] dataset: focuses on social commonsense reasoning regarding the pragmatic implications of characters' intentions, reactions or communal reactions, which can measure LLMs' abilities to handle social reasoning by identifying the most plausible answer.
- **Hellaswag** [110] dataset: a multiple choice benchmark for grounded commonsense reasoning. It includes natural language inference questions, offering a context and the most plausible continuation of a given situation from several candidates, many of which are adversarially generated to be misleading but syntactically plausible.
- **WinoGrande** [111] dataset: a large-scale pronoun resolution benchmark inspired by the Winograd Schema Challenge [112], but adjusted to improve the scale and robustness against the dataset-specific bias. it requires models to resolve ambiguous pronouns using commonsense reasoning rather than simple syntactic cues.
- ARC-Easy and ARC-Challenge (or **ARC-e** and **ARC-c**): two partitions from AI2 Reasoning Challenge [113] dataset, which consists of grade-school level, science example questions. ARC-Easy composes of quizzes solvable with straightforward retrieval or basic reasoning, while ARC-Challenge includes more difficult questions designed to enhance deeper reasoning and knowledge integration.

- Open book question answering - **OBQA** [114] dataset contains multiple-choice elementary-level science questions requiring multi-step reasoning, which makes use of additional common and provided core science facts to get the final answer.

For arithmetic tasks, we concentrate on more difficult datasets with suitable test set size such as GSM8K, along with the more challenging problems of MATH. It could be considered as a standard mathematical reasoning benchmark with complicated inferences compared to MultiArith, AddSub, AquA, SingEq, SVAMP in [43]. In detail, we use these following datasets:

- **MetaMathQA [42]:** the training set. All MetaMathQA questions are bootstrapped from the training sets of GSM8K and Hendrycks MATH by rewriting the samples from multiple viewpoints, but none of the augmented data is from the testing set. The dataset involves forward reasoning and backward reasoning samples.
- **Hendrycks MATH [38]:** a benchmark of competition-style math problems collected from exams such as AMC and AIME. Each problem includes step-by-step solutions and final answers, covering diverse topics like algebra, geometry, number theory, and probability. Designed to test multi-step reasoning rather than rote formula application, MATH highlights the challenges large models face in genuine mathematical problem solving with explanation.
- **GSM8K [39]:** consists of high-quality grade school math word problems, each paired with a detailed explanation. It was designed as a benchmark for evaluating arithmetic reasoning and problem-solving in language models. Compared to MATH, GSM8K focuses on everyday math problems that require clear logical reasoning and generalization, rather than advanced techniques.

I.2 Experimental descriptions

All fine-tuning and evaluation processes are conducted on a single NVIDIA H200/H100 GPU with 140GB/80GB of memory. All the results reported in this section are averaged over multiple experiments and varying random seeds; the terms "parameters" and "#params" here always refer to the number of trainable parameters. In these cases, all experiments are conducted using `bfloat16` precision in order to reduce memory usage, AdamW [15] as the optimizer, and greedy sampling (vLLM: *temperature* $= 0$) unless otherwise specified in Table 16. In each dataset, batch sizes reported here are the effective training batch sizes, including gradient accumulation steps if any.

Table 16: Default common settings across all experiments, unless otherwise specified.

Hyperparameters	**Values**
Learning scheduler	Cosine
Default optimizer	AdamW
AdamW (β_1, β_2)	(0.9, 0.99)
Weight Decay	0.0
Dropout	0.0
Inference sampling	Greedy

I.2.1 Baselines

We evaluate SaMA performance and compare it to the most common and up-to-date baselines. In this section, we provide a brief description of these baselines appearing in our experiments. In fact, we mostly consider fundamental fine-tuning techniques, along with many LoRA variants due to the advances in recent research works.

Full fine-tuning FFT: initializes the model with pre-trained weights and biases, then updates all parameters during training on a downstream task. For LLMs, this technique requires substantial memory and computational cost, but typically serves as a comparative performance bound of PEFT.

Series [47]: The original bottleneck adapter strategy that sequentially inserts trainable modules – composed of a down-projection, a non-linearity, and an up-projection – after the self-attention and feedforward neural network (FFN) layers, while keeping the pre-trained parameters frozen.

Parallel [56]: A structural variant that places the adapter modules alongside the frozen transformer layers (Attention or FFN) rather than sequentially. This approach maintains the original network depth by processing the input in parallel to the frozen weights and summing the results.

LoRA [7]: freezes the pre-trained weights $\mathbf{W}_0 \in \mathbb{R}^{d\times k}$ and injects trainable rank decomposition matrices $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$, applying the update as $\Delta\mathbf{W} = \mathbf{BA}$. LoRA constrains the low intrinsic rank update $r \ll \min(d, k)$ to significantly reduce the number of trainable parameters.

However, the **rsLoRA** [33] variant justifies this scaling to $\alpha/\sqrt{r}$. This rank-stabilized scaling preserves the update norm as the rank increases, ensuring that the optimization dynamics and optimal hyperparameters remain consistent across different rank choices r.

DoRA [28]: improves upon LoRA by structurally decoupling the weight update into a learnable magnitude vector $\mathbf{m}$ and a directional component updated via low-rank adaptation, $\mathbf{W} + \Delta\mathbf{W}$, which comes with a trade-off with forward and backpropagation graphs during training time.

PiSSA [8]: initializes the low-rank adapter matrices $\mathbf{A}$ and $\mathbf{B}$ using the top-r principal components from the Singular Value Decomposition (SVD) of the original weight matrix $\mathbf{W}_0$, rather than random Gaussian initialization.

MiLoRA [64]: leverages SVD of pretrained weight matrices. Unlike LoRA and PiSSA, MiLoRA selectively updates the *minor components* – those with the smallest singular values – while preserving the major components of foundation weight matrices that encode core pretrained knowledge.

KaSA [17]: Unlike PiSSA and MiLORA, which use SVD merely for initialization, KaSA first truncates minor singular values to remove noisy knowledge. It then reparameterizes task-specific updates using learnable singular values to dynamically activate the most relevant components.

LoRA-Pro [36]: An optimization-centric variant that mathematically aligns the gradient updates. LoRA-Pro introduces a closed-form gradient correction ensuring that the weight update ΔW closely approximates the projection of the FFT gradient onto the low-rank subspace defined by r.

HiRA [19]: A high-rank adaptation method that constructs the weight update ΔW using the Hadamard product (element-wise multiplication – denoted as the operator $\odot$) of constituent matrices, rather than the standard matrix multiplication used in LoRA: $\Delta\mathbf{W} = (\mathbf{B}_1\mathbf{A}_1) \odot \mathbf{W}_0$.

ABBA [35]: An alternative to HiRA [19] that reparameterizes the weight update ΔW as the Hadamard product of two independent adapters: $\Delta\mathbf{W} = (\mathbf{B}_1\mathbf{A}_1) \odot (\mathbf{B}_2\mathbf{A}_2)$. By decoupling the update from W_0, ABBA allows the effective rank of the adaptation to reach up to $r_1 r_2$.

SMT [29]: constrains weight updates to a specific sparse or block-diagonal structure, trading off theoretical expressivity for computational and memory efficiency. The primary criterion for block preference is the gradient score during the first phase of warm-up.

QuanTA [11]: is a high-rank adaptation method inspired by quantum-style tensor factorization. It applies small 4D tensor interactions across tensorized feature dimensions through a sequence of transformations with $\binom{N}{2} = \frac{N(N-1)}{2}$ factors to enable richer mixing.

S^{2}FT [82]: selects sparsely and computes densely by updating a small subset of attention heads and hidden channels along with utilizing co-permutation to group these parameters into dense submatrices.

LISA [81]: does partial FFT by randomly freezing transformer layers with a given probability, except for embedding layers which always remain trainable to achieve memory efficiency, based on LoRA's skewed weight norm distribution.

Galore [74]: resolves the memory bottleneck in LLM training by projecting the weight gradients into a low-rank subspace, rather than approximating the weight matrices themselves. By dynamically switching these projection subspaces over time, GaLore enables FFT but is more memory-efficient.

DiaBlo [18]: directly updates only the structured diagonal blocks of the pre-trained weight matrices while keeping the off-diagonal blocks frozen. By avoiding matrix multiplication, DiaBlo simplifies the optimization landscape, targeting stable and robust convergence.

FlexLoRA [65]: a dynamic low-rank adaptation framework that simultaneously prunes redundant ranks in less important layers and expands ranks in critical layers under a global budget.

RepLoRA [10]: reparameterizes low-rank matrices as the outputs of lightweight MLPs, motivated by theoretical connections to Mixture of Experts (MoE) models.

RefLoRA [72]: addresses the slow convergence and imbalanced weight updates of standard LoRA. The method dynamically applies the optimal low-rank factorization at each training step by minimizing an upper bound on the loss function.

RandLoRA [51]: enables full-rank weight updates by modeling them as a linear combination of fixed, low-rank random matrices, restricting the optimization solely to their diagonal coefficients.

LaX [98]: establishes residual connections that directly route the intermediate latent features from a preceding low-rank layer into the up-projection process of the current layer.

I.2.2 The DROP dataset

Table 17: Common settings used for main experiments on DROP.

Experiment	Hyperparameters	Values
Training	Modules	(q_proj, v_proj)
	Batch size	4
	Valid batch	16
	Epoch	2.0
	Eval step	25
	Max token length	1536
Inference	LLM max length	1536
	New tokens	50

Table 18: Learning rate and scaling factor s across various Llama2 model scales on the DROP dataset. Symbol $\star$ indicates the configurations from [11].

Model	Configuration	Learning rate	Scaling
Llama2-7B	LoRA$_{r=8}$	1e−3,2e−3,8e−4	1.0
	LoRA$_{r=32}$	1e−3,2e−3,8e−4	1.0
	LoRA$_{r=128}$	1e−3,2e−3,8e−4	1.0
	QuanTA$^{\star}_{\text{16-8-8-4}}$	1e−4	1.0
	QuanTA$^{\star}_{\text{16-16-16}}$	1e−4	1.0
	SaMA $_{2,2}$	1e−3	1.0
	SaMA $_{4,4}$	1e−3	1.0
	SaMA $_{8,8}$	1e−3	1.0
	SaMA $_{16,16}$	1e−3	1.0
	SaMA $_{32,32}$	1e−3	1.0
	SaMA $_{64,64}$	9e−4	1.0
Llama2-13B	SaMA $_{4\text{-}4}$	8e−4	1.0

In Table 17, we show the common setting for DROP experiments along with hyperparameters in Table 18. The Llama2 family is used as the foundation models.

I.2.3 Commonsense reasoning

To verify the effectiveness of our approach on commonsense reasoning downstream applications, we adapt several pretrained models of the Llama family, including Llama-7B, Llama-13B [44], Llama2-7B and 13B [45], Llama3-8B [46]. This paper [43] introduces an integration of 170k samples from 8 different tasks, but we compare with their metadata and realize that there are many duplicates. Therefore, we pre-process the joint COMMONSENSE170K dataset by removing duplicate samples, yielding COMMONSENSE-147K. Then, following common protocols in this domain [43], we first fine-tune the models on the refined training data and evaluate the same adapted models on the full test sets of downstream tasks. Similar to the methodology used for the DROP dataset, we extract a subset of 1000 samples from COMMONSENSE-147K, then optimize the hyperparameters on this validation set. In the end, we select the best checkpoint based on validation set performance to evaluate the performance on the benchmark. The general hyperparameters and adapter configurations are listed in Tables 19 and 20, respectively.

Attention fine-tuning – a comparison with QuanTA The hyperparameters for these experiments are provided in Table 21:

In Table 12, although the total number of trainable parameters in SaMA is slightly lower than that of QuanTA, SaMA still outperforms in most of the global and local metrics. It further consolidates SaMA capacity for fine-tuning on attention modules.

I.2.4 Arithmetic reasoning

For arithmetic reasoning tasks, we fine-tune pretrained models of Mistral-7B [40] and Gemma-2 9B [41] on a 20K-sample subset of MetaMathQA [42], and evaluate their performance on test sets of MATH [38] and GSM8K [39], which require multi-step reasoning – following [35]. The experimental

Table 19: Common hyperparameters used for main experiments on commonsense reasoning tasks.

Experiment	Hyperparameters	Values
Training	Modules	(Q, K, V, Up, Down)
	Batch size	16
	Valid batch	16, 32
	Epoch	2.0
	Eval step	300
	Max sequence length	512
Inference	LLM max length	1024
	New tokens	50

Table 20: Detailed adapter configurations on commonsense reasoning tasks.

Model	Setting	Learning rate	Scaling (s)	Config (n_1, m_2)
Llama-7B	$\text{SaMA}_{4\times4}$	5e−4	1.414	(4,4)
	$\text{SaMA}_{16\times16}$	5e−4	1.414	(16,16)
Llama-13B	$\text{SaMA}_{4\times4}$	5e−4	1.0	(4,4)
	$\text{SaMA}_{16\times16}$	5e−4	1.0	(16,16)
Llama2-7B	$\text{SaMA}_{4\times4}$	1e−3	0.25	(4,4)
	$\text{SaMA}_{16\times16}$	5e−4	1.0	(16,16)
Llama3-8B	$\text{SaMA}_{4\times4}$	2e−4	1.0	(4,4)
	$\text{SaMA}_{16\times16}$	2e−4	1.0	(16,16)

Table 21: Hyperparameters used for QV-attention experiments on commonsense reasoning, other arguments remain the same as the main experiments. Learning rate and scaling factor correspond respectively to each model.

Experiment	Hyperparameters	Values
SaMA	Modules	(q_proj, v_proj)
	Batch size	16
	Valid batch	32
	Eval step	300
	Config (n_1, m_2)	(4,4)
	Learning rate	{8e-4, 5e-4, 1e-3, 8e-4, 2e-4}
	Scaling factor	1.414, 1.0, 0.5, 1.0, 1.0

Table 22: Common settings used for main experiments on MATH.

Experiment	Hyperparameters	Values
Training	Modules	(q_proj, v_proj, k_proj, o_proj, gate_proj, up_proj, down_proj)
	Batch size	4
	Epoch	1
	Max token length	1536
Inference	LLM max length	4096
	New tokens	1536

Table 23: Learning rate and scaling factor s across math models on the MATH tasks.

Model	Configuration	Learning rate	Scaling factor
Mistral-7B	$\text{SaMA}_{8,8}$	5e−4	1.0
	$\text{SaMA}_{16,16}$	5e−4	1.0
Gemma2-9B	$\text{SaMA}_{8,8}$	5e−4	1.0
	$\text{SaMA}_{16,16}$	5e−4	1.0

hyperparameters are provided in Tables 22 and 23. In these experiments, we fine-tune all linear modules in each transformer layer, including attention and MLP projections.

I.2.5 Vision fine-tuning

Table 24 shows hyperparameters used in the experiments with VTAB-1k [100] on Vision Transformers.

Table 24: Implementation details of configurations, optimizer, and hyper-parameters. We select the best hyper-parameters for each task via grid search.

Configuration	Value
Optimizer	AdamW
Learning Rate	$\{0.1, 0.05, 0.02, 0.01, 0.005, 0.002, 0.001\}$
Weight Decay	$\{0.05, 0.01, 0.005, 0.001\}$
Dropout Rate	$\{0.2, 0.4\}$
Batch Size	$\{32\}$
Learning Rate Schedule	Cosine Decay
Training Epochs	$(24, 29)$
Module	All transformer projections
Warmup Epochs	1

I.3 Template of Instruction fine-tuning

Throughout this work, we use task-specific but structurally consistent prompt templates for both fine-tuning and inference. This subsection summarizes the templates used for DROP, commonsense reasoning, and math reasoning.

I.3.1 Prompt template for DROP

```
Below is a passage followed by a corresponding question that describes a task
Write a response that appropriately completes the request with your answer.

### Instruction:
Passage: {passage}
Question: {question}

### Response:
```

Figure 26: DROP dataset - Prompt template.

Similar to Sec. I.3.2, we use this prompt template during fine-tuning and inference of DROP [37] dataset. The dataset consists of "Passage" and "Question" fields. Therefore, in the templates, we detail these headings and contents into the main body part, as in Fig. 26

I.3.2 Commonsense reasoning data and prompt template

As described in Sec. I.2.3, the commonsense reasoning dataset collection includes 8 different sets. Each one in the domain entails distinct tasks, following [43, 115], we create a structured framework that begins the same dialog template for disparate problems across the collection, followed by separately articulating the objective of the task, then the presentation of the relevant content and the corresponding solution.

Figs. 27 and 28 shows the common instructional part of the template, while Table 25 demonstrates the distinct templates following the common part used to gather the commonsense reasoning data samples for fine-tuning.

Table 25: The data template of each dataset used to create commonsense reasoning data samples.

Dataset	Fine-tuning Data Template
BoolQ	Please answer the following question with true or false, question: [QUESTION] Answer format: true/false the correct answer is [ANSWER]
PIQA	Please choose the correct solution to the question: [QUESTION] Solution1: [SOLUTION_1] Solution2: [SOLUTION_2] Answer format: solution1/solution2 the correct answer is [ANSWER]
SIQA	Please choose the correct answer to the question: [QUESTION] Answer1: [ANSWER_1] Answer2: [ANSWER_2] Answer3: [ANSWER_3] Answer format: answer1/answer2/answer3 the correct answer is [ANSWER]
HellaSwag	Please choose the correct ending to complete the given sentence: [ACTIVITY_LABEL]: [CONTEXT] Ending1: [ENDING_1] Ending2: [ENDING_2] Ending3: [ENDING_3] Ending4: [ENDING_4] Answer format: ending1/ending2/ending3/ending4 the correct answer is [ANSWER]
WinoGrande	Please choose the correct answer to fill in the blank to complete the given sentence: [SENTENCE] Option1: [OPTION_1] Option2: [OPTION_2] Answer format: option1/option2 the correct answer is [ANSWER]
ARC-e & ARC-c	Please choose the correct answer to the question: [QUESTION] Answer1: [ANSWER_1] Answer2: [ANSWER_2] Answer3: [ANSWER_3] Answer4: [ANSWER_4] Answer format: answer1/answer2/answer3/answer4 the correct answer is [ANSWER]
OBQA	Please choose the correct answer to the question: [QUESTION] Answer1: [ANSWER_1] Answer2: [ANSWER_2] Answer3: [ANSWER_3] Answer4: [ANSWER_4] Answer format: answer1/answer2/answer3/answer4 the correct answer is [ANSWER]

```
Below is an instruction that describes a task, paired with an input that
provides further context.
Write a response that appropriately completes the request.

### Instruction:
{instruction}

### Input:
{input}

### Response:
```

Figure 27: Commonsense reasoning - Prompt template with a non-empty input field.

```
Below is an instruction that describes a task, paired with an input that
provides further context.
Write a response that appropriately completes the request.

### Instruction:
{instruction}

### Response:
```

Figure 28: Commonsense reasoning - Prompt template for examples with an empty input field.

```
Below is an instruction that describes a task.
Write a response that appropriately completes the request.


### Instruction:
{instruction}

# Let's think step by step.
### Response:
```

Figure 29: Math reasoning - Prompt template.

I.3.3 Math reasoning prompt template

Fig. 29 shows the actual prompt template when fine-tuning the model and generating outputs on math datasets [42, 38, 39]

J Broader Impact

This paper presents work that aims to advance the field of machine learning. The proposed method, SaMA, along with Morpho parameterization, are designed to efficiently train LLMs and enhance fine-tuning performance. This will facilitate further research regarding PEFT in Large foundation models in the future. We believe that this work does not cause any significant societal consequences.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We theoretically and empirically verify the claims and contributions made in the abstract and introduction. See the abstract and introduction part.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discussed the limitations in Sec. 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: See Appendix D.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Hyperparameters are in Appendix I. The code is provided in the supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All the code and data are available in the supplementary material.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provided all of our experimental settings in Appendix I.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We conducted extensive hyperparameter searches for all baselines and our proposed method. Experimental results are tested multiple times to ensure stability and reliability. See Appendix I.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Appendix I. Experiments are conducted on a single NVIDIA H100 or H200 GPU.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the code of ethics and confirmed that our research complies with its guidelines.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Appendix J

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper presents work whose goal is to advance the field of LLM optimization. All of the datasets and models are public and we have checked them all. Our paper poses none of the risks mentioned in the question.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited the original papers that produced the assets properly.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: All the code is available in the supplementary material, along with an explicit README.md.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used in our core method development.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.