NORMALIZED LOW-RANK ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

While low-rank adaptation (LoRA) is widely used for parameter-efficient model adaptation, how to regularize its training dynamics for stable and effective optimization remains underexplored. Because LoRA initializes the up-projection to zero, its early optimization dynamics are largely governed by the down-projection. Building on this observation, we introduce Normalized Low-Rank Adaptation (NoRA), a simple yet effective method that normalizes the down-projection matrices during training. We further show that the same normalization can be applied only at initialization, improving standard LoRA without requiring repeated normalization throughout training. Across pretraining, supervised finetuning, and reinforcement learning, NoRA consistently accelerates convergence, improves performance and training stability, and mitigates catastrophic forgetting. These benefits require neither additional trainable parameters nor inference-time computation, making NoRA a simple and broadly applicable enhancement to LoRA.

1 INTRODUCTION

As large language models (LLMs) continue to scale, their training costs grow rapidly, increasing the need for parameter-efficient learning. Among parameter-efficient finetuning (PEFT) approaches, Low-rank adaptation (LoRA) (Hu et al., 2021) has become particularly popular due to its simplicity, efficiency, and strong empirical performance. By representing weight updates through low-rank factorization, LoRA greatly reduces the number of trainable parameters and has been successfully applied to supervised finetuning, pretraining, and reinforcement learning (Wang et al., 2026; Shen et al., 2026). Despite its effectiveness, LoRA's optimization dynamics remain poorly understood. Standard LoRA parameterizes $\Delta \boldsymbol{W} = \alpha \boldsymbol{B}\boldsymbol{A}$ with randomly initialized $\boldsymbol{A}$ and zero-initialized $\boldsymbol{B}$. Although this preserves the pretrained initialization, early optimization depends entirely on the random features induced by $\boldsymbol{A}$. This observation motivates the question below:

Does the regularization of the down-projection matrix $\boldsymbol{A}$ provide a useful design dimension for improving LoRA optimization?

Existing work suggests that initialization can substantially affect low-rank optimization. Methods such as PiSSA (Meng et al., 2024) and MiLoRA (Wang et al., 2025) construct the initial low-rank subspace from the spectral structure of pretrained weights and often converge faster than random initialization. However, they require singular-value decomposition and modify the initial decomposition of the pretrained weights, making them less convenient for large-scale training and potentially fragile in reinforcement-learning settings (Yin et al., 2025). Other approaches improve LoRA through modified scaling (Kalajdzievski, 2023), constrained parameterizations (Liu et al., 2024b), or fixed projections (Kopiczko et al., 2024). However, how the regularization of the down-projection matrix affects or improves LoRA's optimization remains poorly understood.

Our investigation begins with the observation that the normalized latent bottleneck used in Multi-head Latent Attention (MLA) substantially stabilizes training and also improves convergence. Similar observations have also been made in multiple training settings (Liu et al., 2017; 2018; Loshchilov et al., 2025). These phenomena suggest that the numerical structure of the normalized latent representation presented to the up-projection can play an important role in stable optimization. Directly introducing latent normalization into LoRA changes the standard linear update from $\boldsymbol{B}\boldsymbol{A}\boldsymbol{x}$ to $\boldsymbol{B} \cdot \mathrm{Norm}(\boldsymbol{A}\boldsymbol{x})$. Because the normalization depends on the input, the resulting transformation is nonlinear with respect to $\boldsymbol{x}$ and can no longer be exactly merged into the pretrained weight matrix

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

like LoRA. Inspired by MLA, we therefore study whether the useful numerical effect of latent normalization can instead be encoded as a regularization of the linear down-projection such that we can have both LoRA's exact mergeability and MLA's favorable optimization benefits.

To this end, we adapt MLA's normalization principle from the latent feature to the low-rank projection itself. Each column of $\boldsymbol{A} \in \mathbb{R}^{r\times k}$ represents how an input coordinate is projected into the r-dimensional latent space. Guided by the normalization dimension in MLA, we propose to normalize these projection vectors along the rank dimension, yielding the linear form $\boldsymbol{B} \cdot \mathrm{Norm}(\boldsymbol{A})\boldsymbol{x}$. We refer to this formulation as Normalized LoRA (NoRA), which constrains the input-to-latent projection magnitudes throughout training while preserving exact weight mergeability. This normalization scheme also ensures that the gradient norm of NoRA can be better aligned with full finetuning, bridging the learnability gap between LoRA and full finetuning.

NoRA requires the down-projection matrix $\boldsymbol{A}$ to be normalized along the output rank dimension. This principle motivates two ways of applying normalization: (1) as a training constraint (*i.e.*, NoRA) and (2) as an initialization strategy (*i.e.*, NoRA-init). In the former, NoRA continuously normalizes the down-projection matrix $\boldsymbol{A}$ along the low-rank dimension r throughout training. In the latter, NoRA-init normalizes $\boldsymbol{A}$ only once at initialization and subsequently optimizes it without constraints during training. For NoRA-init, we propose two initialization variants: (1) random normalization, where $\boldsymbol{A}$ is randomly initialized and then normalized; and (2) Block Identity Matrix Initialization (BIMI), where $\boldsymbol{A}$ is constructed as a block-identity matrix that inherently satisfies the rank-dimensional normalization requirement. NoRA improves training by removing undesirable scale imbalance across the low-rank dimensions, providing a better-conditioned parameterization than standard LoRA.

We evaluate both NoRA and NoRA-init across pretraining, supervised finetuning, and reinforcement learning with verifiable rewards. Across a diverse range of base models, tasks, and training configurations, both methods consistently accelerate convergence, improve training stability, and enhance downstream performance. They also mitigate catastrophic forgetting during supervised adaptation and remain robust in reinforcement-learning settings where other spectral initialization methods can be fragile. In general, we can conclude that these results establish rank-dimension normalization of the down-projection as an important design principle for improving LoRA optimization. Our contributions are summarized as follows:

- We identify the down-projection matrix $\boldsymbol{A}$ as a previously underexplored yet important design dimension in LoRA, showing that its magnitude can play a critical role in shaping optimization dynamics and ultimately improving training efficiency and downstream performance.
- We propose NoRA, which normalizes the down-projection matrix $\boldsymbol{A}$ along the rank dimension in the LoRA parameterization. We further introduce NoRA-init, an initialization-only variant that applies this normalization at initialization and requires no re-normalization during training, enabling seamless performance improvements without modifying the optimization procedure.
- Extensive experiments across pretraining, SFT, and RLVR demonstrate NoRA's improved convergence, stability, downstream performance, and resistance to catastrophic forgetting.

2 PRELIMINARIES

LoRA parameterizes the weight update of a pretrained matrix as

$$\Delta \boldsymbol{W} = \boldsymbol{W} - \boldsymbol{W}_0 = \alpha \boldsymbol{B}\boldsymbol{A}, \tag{1}$$

where $\boldsymbol{A} \in \mathbb{R}^{r\times k}$, $\boldsymbol{B} \in \mathbb{R}^{d\times r}$, and $r \ll \min(d, k)$. Standard LoRA initializes $\boldsymbol{B}^{(0)} = \boldsymbol{0}$ to preserve the pretrained model. Let $\boldsymbol{G} = \partial\mathcal{L}/\partial\boldsymbol{W}$ denote the gradient of the corresponding full weight matrix. At initialization, we have the following gradients with respect to $\boldsymbol{B}$ and $\boldsymbol{A}$:

$$\left.\frac{\partial \mathcal{L}}{\partial \boldsymbol{B}}\right|_{t=0} = \alpha \boldsymbol{G}(\boldsymbol{A}^{(0)})^\top, \qquad \left.\frac{\partial \mathcal{L}}{\partial \boldsymbol{A}}\right|_{t=0} = \boldsymbol{0}. \tag{2}$$

Therefore, the earliest optimization of LoRA is entirely determined by the latent representation induced by the initial projection $\boldsymbol{A}^{(0)}$.

Motivated by this observation, we consider a unified latent-feature formulation:

$$\Delta \boldsymbol{y} = \alpha \boldsymbol{B}\boldsymbol{\phi}(\boldsymbol{x}), \tag{3}$$

Method	Forward Computation	Initialization / Prior
Full-Parameter and Standard Low-Rank Adaptation		
Full Finetuning	$y = \boldsymbol{W}_0 x$	N/A
LoRA (Hu et al., 2021)	$y = \boldsymbol{W}_0 x + \boldsymbol{B}\boldsymbol{A}x$	$\boldsymbol{A} \sim \mathcal{N}(0, \sigma^2),\ \boldsymbol{B} = 0$
Optimization-Strategy Variants		
LoRA+ (Hayou et al., 2024)	$y = \boldsymbol{W}_0 x + \boldsymbol{B}\boldsymbol{A}x$	$\eta_{\boldsymbol{B}} = \lambda \eta_{\boldsymbol{A}}$
Structured or Constrained Low-Rank Parameterizations		
DoRA (Liu et al., 2024b)	$y = \boldsymbol{m}\left(\frac{\boldsymbol{W}_0 x + \boldsymbol{B}\boldsymbol{A}x}{\|\boldsymbol{W}_0 + \boldsymbol{B}\boldsymbol{A}\|_c}\right)$	Kaiming init, $\boldsymbol{B} = 0$
MiSS (Kang & Yin, 2026)	$y = \boldsymbol{W}_0 x + \mathrm{expand}(\boldsymbol{D})x$	Fixed structured projection
AdaLoRA (Zhang et al., 2023b)	$y = \boldsymbol{W}_0 x + \boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{Q}x$	$\boldsymbol{\Lambda} = 0,\ \boldsymbol{P}, \boldsymbol{Q} \sim \mathcal{N}(0, \sigma^2)$
LoRA-FA (Zhang et al., 2023a)	$y = \boldsymbol{W}_0 x + \boldsymbol{B}\boldsymbol{A}x$	Frozen random $\boldsymbol{A},\ \boldsymbol{B} = 0$
VeRA (Ye et al., 2025)	$y = \boldsymbol{W}_0 x + \boldsymbol{\Lambda}_b \boldsymbol{B} \boldsymbol{\Lambda}_d \boldsymbol{A}x$	Frozen random $\boldsymbol{A}, \boldsymbol{B}$
Initialization-Based Low-Rank Adaptation		
PiSSA (Meng et al., 2024)	$y = (\boldsymbol{W}_0 - \boldsymbol{B}\boldsymbol{A})x + \boldsymbol{B}\boldsymbol{A}x$	Top-r SVD initialization
MiLoRA (Wang et al., 2025)	$y = (\boldsymbol{W}_0 - \boldsymbol{B}\boldsymbol{A})x + \boldsymbol{B}\boldsymbol{A}x$	Minor-component SVD init
NoRA-init	$y = \boldsymbol{W}_0 x + \boldsymbol{B}\boldsymbol{A}x$	$\boldsymbol{A} = \mathrm{Norm}(\boldsymbol{A}),\ \boldsymbol{B} = 0$
NoRA	$y = \boldsymbol{W}_0 x + \boldsymbol{B}\,\mathrm{Norm}(\boldsymbol{A})\boldsymbol{x}$	$\boldsymbol{B} = 0$

Table 1: Overview of representative PEFT methods categorized by their primary mechanism, including forward parameterization and initialization strategy.

where $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^r$ denotes the latent feature presented to the up-projection. Existing low-rank methods can be interpreted as constructing this latent feature in different ways:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{cases} \boldsymbol{A}\boldsymbol{x}, & \text{random projection (LoRA)}, \\ \boldsymbol{A}_{\text{init}}\boldsymbol{x}, & \text{structured projection (e.g., PiSSA, MiLoRA)}, \\ \mathrm{Norm}(\boldsymbol{A}\boldsymbol{x}), & \text{normalized projection (e.g., MLA)}. \end{cases} \tag{4}$$

Although these approaches adopt different implementations, they can all be viewed as designing the latent feature presented to the up-projection. This latent-projection perspective motivates us to revisit how to regularize the latent projection without losing the exact mergeability of LoRA.

3 NoRA: Normalized Low-Rank Adaptation

3.1 From Normalized Latent Features to Normalized Low-Rank Projection

The normalized latent projection in MLA suggests that the representation presented to the up-projection plays an important role in training dynamics. Specifically, MLA constructs the low-dimensional latent feature as

$$\boldsymbol{\phi}(\boldsymbol{x}) = \mathrm{Norm}(\boldsymbol{A}\boldsymbol{x}), \tag{5}$$

where the normalization operator explicitly regulates the projected feature before it is mapped back to the output space. Directly introducing this operation into LoRA changes the standard linear update from $\boldsymbol{B}\boldsymbol{A}\boldsymbol{x}$ to $\boldsymbol{B}\mathrm{Norm}(\boldsymbol{A}\boldsymbol{x})$. Since the normalization depends on the input, this transformation becomes nonlinear with respect to $\boldsymbol{x}$ and can no longer be exactly merged into the pretrained weight matrix. We therefore seek to transfer the scale-control effect from the latent feature to the down-projection matrix itself. Modern Transformer architectures are typically pre-normalized, such that the magnitude of the input $\boldsymbol{x}$ to each linear layer is already well controlled by LayerNorm. In this setting, variations in the column norms of $\boldsymbol{A}$ introduce additional scale imbalance in how different input coordinates are projected into the low-rank latent space. We therefore normalize each column of $\boldsymbol{A}$ along the LoRA rank dimension. For $\boldsymbol{A} \in \mathbb{R}^{r \times k}$, we write

$$\boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_k], \qquad \boldsymbol{a}_j \in \mathbb{R}^r, \tag{6}$$

where $\boldsymbol{a}_j$ denotes the latent projection vector associated with the j-th input coordinate. We define

$$\mathrm{Norm}(\boldsymbol{A}) = \left[\frac{\boldsymbol{a}_1}{\max(\|\boldsymbol{a}_1\|_2, \epsilon)}, \ldots, \frac{\boldsymbol{a}_k}{\max(\|\boldsymbol{a}_k\|_2, \epsilon)}\right], \tag{7}$$

where ϵ is a small constant for numerical stability. Thus, we have that

$$\left\|[\mathrm{Norm}(\boldsymbol{A})]_{:,j}\right\|_2 = 1, \qquad j = 1, \ldots, k. \tag{8}$$

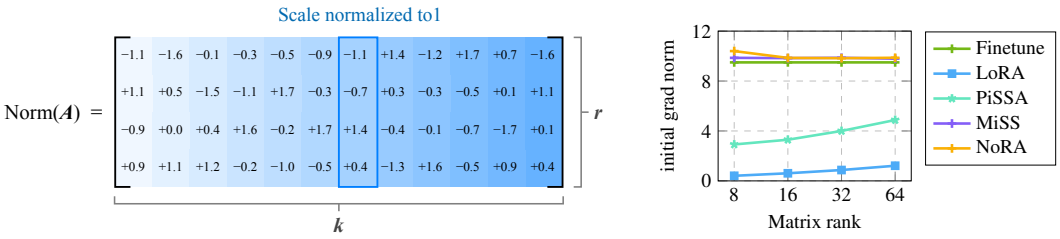

Figure 1: Illustration and optimization behavior NoRA. **Left:** Each input-to-latent projection vector is normalized along the LoRA rank dimension. **Right:** Gradient-norm dynamics of different initialization methods and LoRA ranks on LLaMA-3.2-3B trained on the Math dataset.

Based on this rank-dimension normalization, we introduce normalized low-rank adaptation, which parameterizes the low-rank update as the following form:

$$\Delta \boldsymbol{y} = \alpha \boldsymbol{B}\mathrm{Norm}(\boldsymbol{A})\boldsymbol{x}. \tag{9}$$

Unlike latent-feature normalization in MLA, $\mathrm{Norm}(\cdot)$ depends only on the LoRA parameters rather than the input. NoRA therefore imposes a normalization constraint on the down-projection throughout training while remaining linear with respect to $\boldsymbol{x}$. After training, the normalized down-projection can be absorbed into the low-rank update, preserving exact weight mergeability. In this way, every input coordinate is continuously projected through a unit-norm latent direction, removing arbitrary column-wise scale variation in $\boldsymbol{A}$. The resulting transition can be summarized as

$$\underbrace{\boldsymbol{B}\mathrm{Norm}(\boldsymbol{A}\boldsymbol{x})}_{\textbf{MLA: input-dependent latent normalization}} \quad \rightarrow \quad \underbrace{\boldsymbol{B}\mathrm{Norm}(\boldsymbol{A})\boldsymbol{x}}_{\textbf{NoRA: normalized low-rank projection}} . \tag{10}$$

Our initialization and gradient-norm analyses further suggest that much of the benefit of NoRA is already established at the beginning of optimization. This motivates a simpler initialization-only variant, which we denote as NoRA-init. Specifically, we initialize

$$\boldsymbol{A}^{(0)} = \mathrm{Norm}(\boldsymbol{A}_{\text{init}}), \qquad \boldsymbol{B}^{(0)} = \boldsymbol{0}, \tag{11}$$

and subsequently optimize $\boldsymbol{A}$ and $\boldsymbol{B}$ using the standard LoRA parameterization without performing normalization throughout training. $\boldsymbol{A}_{\text{init}}$ denotes the random initialization in LoRA. Despite removing the persistent constraint during training, NoRA-Init retains most of the optimization benefit of NoRA in our experiments. This observation suggests that controlling the scale of the input-to-latent projection at initialization is particularly important for the early optimization of low-rank adaptation.

3.2 Connection to MiSS and Block Identity Matrix Initialization

Having established the effectiveness of rank-dimension normalization, we further investigate alternative approaches for achieving this form of normalization. MiSS (Kang & Yin, 2026) is a recent PEFT method that improves parameter efficiency through a matrix-shard sharing strategy. Consider an input $\boldsymbol{x} \in \mathbb{R}^k$ with $k = br$, partitioned into b blocks $\boldsymbol{x}^{(i)} \in \mathbb{R}^r$. The MiSS update is written as

$$\Delta \boldsymbol{y}_{\text{MiSS}} = \Delta \boldsymbol{W}\boldsymbol{x} = \alpha \cdot \mathrm{expand}(\boldsymbol{B})\boldsymbol{x} = \alpha \boldsymbol{B} \sum_{i=1}^{b} \boldsymbol{x}^{(i)} = \alpha \boldsymbol{B} \underbrace{[\boldsymbol{I}_r, \ldots, \boldsymbol{I}_r]}_{\boldsymbol{A}_{\text{fix}}} \boldsymbol{x}. \tag{12}$$

Thus, MiSS is equivalent to a LoRA-style update with a fixed down-projection $\boldsymbol{A}_{\text{fix}}$ composed of repeated identity matrices. Importantly, each column of this projection has unit norm:

$$\|(\boldsymbol{A}_{\text{fix}})_{:,j}\|_2 = 1, \qquad \forall j. \tag{13}$$

MiSS therefore can be interpreted as a special case of NoRA, as its down-projection matrix is normalized by a block-identity construction (*i.e.*, $\boldsymbol{A}$ is fixed to the constant $\boldsymbol{A}_{\text{fix}}$). As shown in Figure 1, NoRA produces substantially stronger early gradients than standard LoRA, with gradient norms approaching those of full finetuning. Interestingly, MiSS also exhibits a highly similar pattern. These results well justifies the empirical effectiveness of NoRA and its special case MiSS.

Motivated by the connection to MiSS, we introduce Block Identity Matrix Initialization (BIMI) as a simple structured realization of NoRA-init. Rather than fixing the block-identity projection

throughout training, BIMI uses it only to initialize the LoRA down-projection matrix and allows it to remain fully trainable afterward. Specifically, for $k = br + q$, where $0 \le q < r$, we initialize

$$\boldsymbol{A}_{\text{bimi}} = \left[\underbrace{\boldsymbol{I}_r, \boldsymbol{I}_r, \ldots, \boldsymbol{I}_r}_{b \text{ blocks}}, \boldsymbol{E}_q \right] \in \mathbb{R}^{r \times k}, \tag{14}$$

where $\boldsymbol{E}_q \in \mathbb{R}^{r \times q}$ contains the first q columns of $\boldsymbol{I}_r$ and is omitted when $q = 0$. By construction, we have $\|(\boldsymbol{A}_{\text{bimi}})_{:,j}\|_2 = 1, \forall j,$, making BIMI directly a special case of NoRA-init. The resulting update retains the standard LoRA form, $\Delta \boldsymbol{y} = \alpha \boldsymbol{B}\boldsymbol{A}\boldsymbol{x}$, with both $\boldsymbol{A}$ and $\boldsymbol{B}$ optimized during training.

3.3 Why Does NoRA Work? A Preconditioning Perspective

The early optimization of LoRA is dictated by the initialization of the down-projection matrix $\boldsymbol{A}$. This is because LoRA can be viewed as performing full-finetuning gradient descent under a hidden preconditioner that is determined by $\boldsymbol{A}$ and acts on the input coordinates. Let $\boldsymbol{G} = \partial \mathcal{L}/\partial \boldsymbol{W} \in \mathbb{R}^{d \times k}$ be the full-finetuning gradient. A gradient step of size η on $\boldsymbol{B}$ at initialization is $\Delta \boldsymbol{B} = -\eta \alpha \boldsymbol{G}\boldsymbol{A}^\top$, while $\boldsymbol{A}$ receives no gradient and stays fixed. The induced change of the merged weight is therefore

$$\Delta \boldsymbol{W} = \alpha \, \Delta \boldsymbol{B} \, \boldsymbol{A} = -\eta \, \boldsymbol{G} \, \boldsymbol{P}, \qquad \boldsymbol{P} = \alpha^2 \boldsymbol{A}^\top \boldsymbol{A} \in \mathbb{R}^{k \times k}. \tag{15}$$

Full finetuning takes the step $\Delta \boldsymbol{W} = -\eta \, \boldsymbol{G}$, *i.e.*, equation 15 with $\boldsymbol{P} = \boldsymbol{I}$. Hence, at initialization, and approximately for as long as $\boldsymbol{B}$ remains small (the neglected term $\alpha \boldsymbol{B} \, \Delta \boldsymbol{A} = -\eta \alpha^2 \boldsymbol{B}\boldsymbol{B}^\top \boldsymbol{G}$ is second order in $\boldsymbol{B}$), LoRA is full finetuning with the gradient right-multiplied by a positive semidefinite matrix of rank r that acts on the input coordinates. Right-multiplication by a $k \times k$ matrix is exactly where curvature-based methods (Martens & Grosse, 2015) place an input-side preconditioner. Moreover, we observe that column norms are per-coordinate learning rates:

$$\Delta \boldsymbol{W}_{:,j} = -\eta \Big[\underbrace{\alpha^2 \|\boldsymbol{a}_j\|_2^2 \, \boldsymbol{G}_{:,j}}_{\text{coordinate } j\text{'s own update}} + \underbrace{\sum\nolimits_{i \neq j} \alpha^2 (\boldsymbol{a}_i^\top \boldsymbol{a}_j) \, \boldsymbol{G}_{:,i}}_{\text{crosstalk from the rank bottleneck}} \Big]. \tag{16}$$

The first term is the full-finetuning update of the weight column with respect to input coordinate j, scaled by the squared length of its latent $\boldsymbol{a}_j$. The second term is crosstalk between input coordinates induced by the rank bottleneck, and its coefficients are inner products of latent directions.

Random initialization of $\boldsymbol{A}$ makes learning rates unbalanced at each coordinate. For example, when $\boldsymbol{A}^{(0)}$ has zero-mean i.i.d. entries of variance $\sigma^2 \propto 1/k$, $\mathbb{E}\|\boldsymbol{a}_j\|_2^2 = r\sigma^2 \propto r/k \ll 1$. Therefore, at any moderate α, every learning rate is far too small, so gradient flows through the adapter at a small fraction of the full-finetuning rate. This is empirically verified by the small gradient norm of LoRA in Figure 1. The learning rates are also unbalanced. $\|\boldsymbol{a}_j\|_2^2$ fluctuates around its mean with relative spread of order $1/\sqrt{r}$, so at small rank each coordinate gets its learning rate randomly, unrelated to data or curvature. A preconditioner should remove distortion, not add its own.

Normalization is exactly the fix. Setting every $\|\boldsymbol{a}_j\|_2 = 1$ at unit scaling (rank-dimension normalization in NoRA), $\alpha = 1$ (equivalently $\alpha = r$ under the α/r implementation convention), fixes the problems. Specifically, this leads to $\text{Diag}(\boldsymbol{P}) = \boldsymbol{I}$ deterministically, $\mathbb{E}[\boldsymbol{P}] = \boldsymbol{I}$ over the random directions, and the adapter's gradient norm can match full finetuning's gradient norm in expectation, independently of r. Therefore, we can get the flat NoRA curve in Figure 1. In contrast, row normalization (Norm_k) leaves $\text{Diag}(\boldsymbol{P})$ random of order r/k, and brings no empirical gain (see Table 3). Moreover, BIMI uses unit basis vectors as columns (the same diagonal through a completely different crosstalk pattern) and performs similarly to rank-dimension normalization of random initializations (see Table 3), which points to the diagonal of $\boldsymbol{P}$ as the decisive quantity.

Initialization does most of the work; the constraint adds stability. Since $\boldsymbol{A}$'s gradient, $\alpha \boldsymbol{B}^\top \boldsymbol{G}$, vanishes with $\boldsymbol{B}$, $\boldsymbol{A}^{(0)}$ alone fixes the learning rates and the input subspace during the decisive early phase, correcting the rates once at initialization thus captures most of the benefit. This is exactly what NoRA-init does. Performing $\text{Norm}(\boldsymbol{A})$ in the forward pass adds two things: (1) the loss becomes invariant to each column's scale, so gradients are tangential and, under gradient descent, norms can only grow while the effective step anneals automatically, as in weight-normalized training (Liu et al., 2017; Salimans & Kingma, 2016; Liu et al., 2018); and (2) $\text{Diag}(\boldsymbol{P}) = \boldsymbol{I}$ is enforced for the whole run on a compact set of directions, so the latent scale can neither collapse nor explode and the update stays linear in $\boldsymbol{x}$ and can be merged exactly. To summarize, NoRA sets the diagonal gains of LoRA's hidden preconditioner to one, and keeps them throughout training.

Stage	Repository	Model	Train Data	Benchmark
Pretrain	FLA (Yang & Zhang, 2024)	MLA (Liu et al., 2024a) MHA (Vaswani et al., 2017)	SlimPajama (Soboleva et al., 2023)	LAMBADA (Paperno et al., 2016) WikiText (Merity et al., 2016) Arc-Easy/Arc-Challenge (Clark et al., 2018) HellaSwag (Zellers et al., 2019) PIQA (Bisk et al., 2020) OpenBookQA (Mihaylov et al., 2018) WinoGrande (Sakaguchi et al., 2021)
SFT	PEFT-Arena (Huang et al., 2026)	Llama-3.2-3B (Grattafiori et al., 2024)	MetaMath (Yu et al., 2024) CodeFeedback (Zheng et al., 2024)	**Math:** GSM8k (Cobbe et al., 2021) Math500 (Yu et al., 2024) **Code:** HumanEval (Chen et al., 2021) Mbpp (Austin et al., 2021)
RL	PeRL (Yin et al., 2025)	DeepSeek-R1-Distill-Qwen1.5B (Liu et al., 2024a)	open-r1/DAPO-Math-17k-Processed (Yu et al., 2026)	AIME24 (Zhang & Math-AI, 2025) (Avg@32) AIME25 (Zhang & Math-AI, 2025) (Avg@32) MATH500 (Lightman et al., 2024) (Avg@4) Minerva (Lewkowycz et al., 2022) (Avg@4) AMC (Li et al., 2024) (Avg@32) HMMT (Balunović et al., 2025) (Avg@32)

Table 2: Overview of all the training settings in our experiments.

4 Experiments and Results

To thoroughly evaluate our method, we conduct experiments across pretraining, supervised finetuning (SFT), and reinforcement learning. We further perform ablation studies to investigate the effect of the normalization dimension and initialization distribution. Evaluation settings are in Table 2.

Ablation Study. We investigate the effect of the normalization dimension of the down-projection matrix $\boldsymbol{A}$ by comparing column-wise and row-wise normalization, and further evaluate the effectiveness of NoRA across different random or deterministic initialization schemes.

Pretraining. We begin with controlled studies on the MLA architecture under the L24-D1024 setting to examine the effect of intermediate normalization within low-rank latent representations. Building on these observations, we further evaluate NoRA on standard multi-head attention (MHA), where all attention projection layers are adapted with NoRA.

Supervised finetuning. We evaluate NoRA on Llama 3.2-3B across a diverse set of downstream tasks, including mathematical reasoning and code generation, as well as its robustness against catastrophic forgetting (Huang et al., 2026). More detailed settings are given in Appendix B.

Reinforcement learning with verifiable rewards. To assess its effectiveness in post-training optimization, we additionally evaluate NoRA under reinforcement learning using the ReRL framework.

4.1 Effect of the Normalization Dimension

We first study whether the effect of the normalization dimension under different initialization schemes. Given $\boldsymbol{A} \in \mathbb{R}^{r\times k}$, Norm_k normalizes each row, while Norm_r normalizes each column $\boldsymbol{A}_{:,j} \in \mathbb{R}^r$. As shown in Table 3, Norm_k provides little improvement, whereas Norm_k consistently improves Random Uniform, Gaussian, and Kaiming Uniform initializations. After applying NoRA, all three initialization schemes achieve similar performance, indicating that its benefit is largely independent of the initialization distribution. BIMI achieves comparable performance and satisfies NoRA by construction, serving as a structured and deterministic instance of NoRA.

Initialization	Method	GSM8K	Math	Avg.
$\mathcal{U}(-1/\sqrt{k}, 1/\sqrt{k})$	None	47.68	10.86	29.27
	Norm_k	48.67	10.82	29.75
	Norm_r	**60.12**	14.20	**37.16**
$\mathcal{N}(0, 1/r^2)$	None	50.41	11.68	31.05
	Norm_k	49.73	11.34	30.54
	Norm_r	<u>59.96</u>	13.93	36.95
$\mathcal{U}(-1, 1)$	None	48.19	11.02	29.61
	Norm_k	48.29	10.86	29.58
	Norm_r	58.98	**14.34**	36.66
$[\boldsymbol{I}_r, \ldots, \boldsymbol{I}_r]$	BIMI	59.89	<u>14.24</u>	<u>37.07</u>

Table 3: Effect of different normalization dimensions under supervised finetuning. The average is computed over accuracies on GSM8K and Math.

4.2 LLM Pretraining

We first conduct LLM pretraining experiments using the standard MLA architecture. Normalizing the latent representation $\boldsymbol{Ax}$ consistently improves model performance and accelerates convergence

Model & Scale	Parameters	Lamb.† ppl↓	Wiki. ppl↓	Lamb. acc↑	ARC_e acc↑	ARC_c acc_n↑	Hella. acc_n↑	PIQA acc↑	Wino. acc↑	OBQA acc↑	Avg. acc↑
MLA with 10B training tokens and 0.5M batchsize tokens											
$B\text{Norm}(Ax)$	342.4M	**48.73**	**32.24**	**30.62**	**56.99**	**27.56**	**36.26**	63.49	51.30	**22.00**	**41.17**
BAx	342.4M	61.83	32.89	28.70	54.63	26.02	35.88	63.66	51.62	**22.00**	40.36
$BA_{\text{NoRA-init}}x$	342.4M	50.77	32.42	29.87	54.04	26.37	36.11	**63.76**	**52.72**	21.60	40.64
MHA with 10B training tokens and 0.5M batchsize tokens											
Wx	348.7M	**47.95**	**31.31**	**30.55**	**54.59**	**26.45**	**36.95**	**65.07**	**51.22**	**22.60**	**41.06**
BAx	289.3M	-	-	0.00	25.17	28.07	25.97	49.78	49.49	16.20	27.81
$BA_{\text{NoRA-init}}x$	289.3M	63.45	34.39	28.29	54.38	26.02	35.10	63.49	50.91	19.60	39.69

Table 4: Zero-shot performance of 340M models trained on SlimPajama (Soboleva et al., 2023). Commonsense reasoning tasks are evaluated with lm-evaluation-harness (Gao et al., 2024); the recall-intensive task follows prefix-linear-attention (Arora et al., 2024) with 2K input tokens. "-" indicates a collapse in perplexity.

Method	Supervised Finetuning						Forgetting			
	Trainable	GSM8K	Math	HumanEval	MBPP	Avg	MMLU	AGIEval	ARC-C	Avg Δ
Base Model	-	–	–	–	–	–	$55.10_{+0.00}$	$23.99_{+0.00}$	$42.92_{+0.00}$	0.00
Full	3B	**65.12**	**17.96**	36.15	49.05	42.07	$54.17_{-0.93}$	**26.25**$_{+2.26}$	$41.64_{-1.28}$	+0.02
PiSSA	48.6M	54.66	12.08	39.00	55.30	40.26	$54.43_{-0.67}$	$23.99_{+0.00}$	$42.75_{-0.17}$	-0.28
OFT	53.7M	56.10	13.02	38.40	54.20	40.43	**54.72**$_{-0.38}$	$25.23_{+1.25}$	**43.60**$_{+0.68}$	**+0.52**
RSLoRA	48.6M	57.05	12.32	39.65	**56.10**	41.28	$54.07_{-1.03}$	$24.58_{+0.60}$	$41.81_{-1.11}$	-0.51
MiSS	49.5M	60.80	14.60	**39.60**	55.80	**42.70**	$53.40_{-1.50}$	$24.33_{+0.34}$	$41.98_{-0.94}$	-0.70
LoRA-style Parameterization										
LoRA	48.6M	50.94	10.38	37.20	53.20	37.93	**54.27**$_{-0.83}$	$24.40_{+0.42}$	$41.64_{-1.28}$	-0.56
NoRA-init	48.6M	59.89	14.24	39.60	**55.80**	42.38	$54.09_{-1.01}$	$25.10_{+1.11}$	$42.24_{-0.68}$	-0.19
NoRA	48.6M	**61.63**	**14.46**	**42.10**	55.30	**43.37**	$54.00_{-1.10}$	**25.32**$_{+1.33}$	**42.75**$_{-0.17}$	**+0.02**
DoRA-style Parameterization										
DoRA	49.4M	51.63	11.36	37.80	52.40	38.30	**54.20**$_{-0.90}$	$24.09_{+0.10}$	$41.81_{-1.11}$	-0.64
NoRA-init	49.4M	59.59	14.10	38.70	53.20	41.40	$53.65_{-1.45}$	**24.84**$_{+0.86}$	**42.32**$_{-0.60}$	**-0.40**
NoRA	49.4M	**61.33**	**14.44**	**42.10**	**55.60**	**43.34**	$54.00_{-1.10}$	$24.59_{+0.58}$	$42.15_{-0.77}$	-0.43

Table 5: Comparison of PEFT methods under supervised finetuning and retained benchmark settings.

during pretraining. Motivated by these results, we extend the same normalization principle to the linear low-rank formulation by applying normalization to the down-projection $\boldsymbol{A}$, yielding NoRA. To evaluate the effectiveness of NoRA for LLM pretraining, we adopt NoRA-init and compare it with the standard low-rank parameterization. As shown in Table 4, NoRA-init achieves faster convergence and better downstream performance. Although its performance remains slightly below that of directly normalizing $\boldsymbol{Ax}$, NoRA-init preserves the linear structure $\boldsymbol{BAx}$, thereby retaining the parameter mergeability and deployment efficiency of conventional low-rank parameterizations.

We further evaluate NoRA-init under the standard MHA architecture for LLM pretraining. Specifically, we parameterize all linear projections in the attention module, including $\boldsymbol{q}_{\text{proj}}$, $\boldsymbol{k}_{\text{proj}}$, $\boldsymbol{v}_{\text{proj}}$, and $\boldsymbol{o}_{\text{proj}}$, using low-rank factors with rank $r = 64$. As shown in Table 4, the standard low-rank parameterization exhibits a severe performance collapse on LAMBADA, along with substantial degradation on several other benchmarks. An examination of the optimization dynamics reveals that its gradient norms diminish to abnormally low levels during training, suggesting insufficient optimization. In contrast, NoRA-init achieves significantly better performance than standard low-rank parameterization and preserves well-scaled gradient magnitudes throughout training, thereby avoiding such optimization collapse and yielding more stable loss trajectories and downstream performance.

This observation is related to the capacity-limited behavior of low-rank adaptation discussed in Schulman & Lab (2025), where insufficient low-rank capacity can largely constrain the flexibility of optimization trajectory. Our results further show that capacity alone does not fully determine optimization behavior. Under the same rank constraint, modifying the initialization of the low-rank projection substantially changes the training dynamics. This suggests that controlling the scale and structure of the input-to-latent projection provides a more favorable optimization geometry under limited-rank parameterizations, thereby alleviating performance degradation in pretraining.

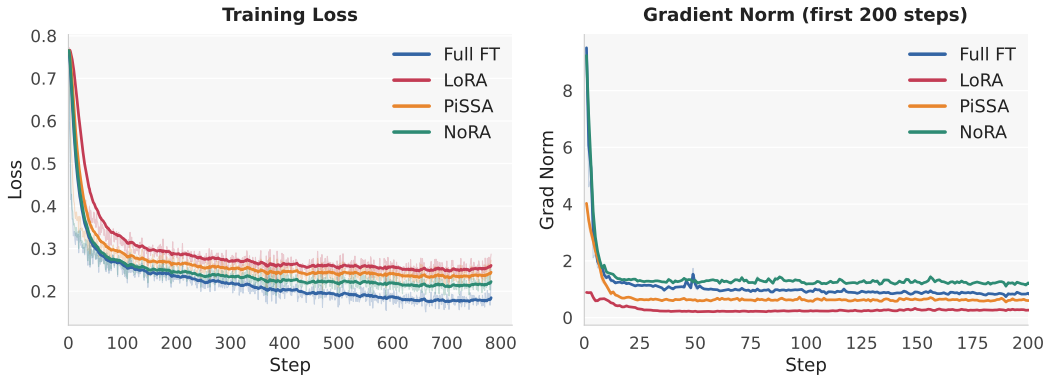

Figure 2: Training dynamics comparison of full finetuning, LoRA, PiSSA, and NoRA on the SFT task.

4.3 Supervised Finetuning

Table 5 compares NoRA with representative PEFT methods under supervised finetuning. NoRA achieves the best overall SFT performance among the compared methods, improving the average score from 37.93 with standard LoRA to 43.37, a gain of 5.44 points. In particular, NoRA substantially improves performance on GSM8K and HumanEval, reaching 61.63 and 42.10, respectively, while also outperforming PiSSA (Meng et al., 2024), OFT (Qiu et al., 2023; 2025; Liu et al., 2024c), RSLoRA (Kalajdzievski, 2023), and MiSS (Kang & Yin, 2026) in average performance. These results demonstrate the effectiveness of rank-dimension normalization for improving LoRA.

We further compare NoRA with its initialization-only variant, NoRA-init. NoRA-init applies normalization constraint only to the initial down-projection and then follows standard LoRA training, whereas NoRA continuously enforces the normalization constraint throughout training. NoRA-init already improves the SFT average from 37.93 to 42.38, capturing a large portion of the gain achieved by NoRA. With the normalization constraint maintained throughout training, NoRA further improves the average score to 43.37. The comparison indicates that controlling the input-to-latent projection magnitude at initialization accounts for a large portion of the gain, while persistent normalization provides additional improvements. Comparison of training dynamics is given in Figure 2. NoRA shows stronger learnability than LoRA without introducing extra parameters.

NoRA also outperforms MiSS, which achieves an average score of 42.70. Recall that MiSS can be reformulated as a fixed block-identity down-projection satisfying the normalization constraint. The strong performance of both MiSS and NoRA-init therefore provides further evidence that normalized low-rank projections play an important role in their optimization. Notably, NoRA-init achieves comparable performance to MiSS while keeping the down-projection fully trainable, avoiding the constraint of fixing the projection throughout training.

Finally, we examine whether the NoRA principle generalizes beyond the standard LoRA parameterization. We apply Norm_r-based normalization to DoRA and obtain consistent improvements: NoRA-init-DoRA improves the SFT average from 38.30 for DoRA to 41.40. This result demonstrates that the benefit of rank-dimension normalization is not specific to LoRA, but can also be transferred to other low-rank adaptation variants. Together with the improvements observed for LoRA, this suggests that NoRA captures a general optimization principle for low-rank adaptation. NoRA also exhibits favorable knowledge retention, achieving an average change of +0.02 on MMLU, AGIEval, and ARC-C, compared with -0.56 for LoRA and -0.70 for MiSS. Thus, NoRA improves adaptation performance while maintaining the pretrained model's retained knowledge.

4.4 RLVR on Mathematical Reasoning

We further evaluate NoRA under reinforcement learning with verifiable rewards (RLVR) on challenging mathematical reasoning benchmarks. As shown in Table 6, standard LoRA improves the overall average score from 41.0 for the base model to 42.8, while NoRA further increases it to 44.4, outperforming the base model and standard LoRA by 3.4 and 1.6 points, respectively. NoRA yields broad improvements across the evaluated benchmarks, with particularly clear gains on AMC,

Method	AIME24@32		AIME25@32		AMC@32		HMMT@32		MATH500@4		Minerva@4		Avg.
	Avg.	Pass	Avg.	Pass	Avg.	Pass	Avg.	Pass	Avg.	Pass	Avg.	Pass	
Base	24.3	70.0	20.4	**53.3**	64.8	95.0	9.9	**36.7**	80.9	**92.8**	27.8	43.0	41.0
MiLoRA	4.2	6.7	0.0	0.0	19.6	47.5	0.0	0.0	44.5	63.4	11.7	19.9	18.0
PiSSA	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.6	1.0	0.1	0.4	0.2
LoRA	**28.0**	60.0	23.1	**53.3**	68.0	**97.5**	**10.3**	35.0	81.3	92.2	30.1	46.3	42.8
NoRA	26.7	**66.7**	**23.5**	46.7	**72.7**	**97.5**	**10.3**	35.0	**84.5**	**92.8**	**32.0**	**47.8**	**44.4**

Table 6: Comparison of accuracy and pass rate on the RLVR task. All numbers are reported in percentages.

MATH500, and Minerva. These results suggest that rank-dimension normalization remains effective under RLVR and facilitates more effective optimization of low-rank adapters.

A notable difference emerges for initialization methods based on pretrained-weight spectral decomposition. Although PiSSA (Meng et al., 2024) and MiLoRA (Wang et al., 2025) are effective under supervised finetuning, their performance degrades substantially in the RLVR setting. This behavior is consistent with prior observations that spectral initialization methods can suffer from optimization instability during reinforcement learning (Yin et al., 2025). In contrast, NoRA remains stable and consistently improves the overall performance of standard LoRA without relying on singular-value decomposition of the pretrained weights. These results highlight the robustness of NoRA across different optimization regimes. Unlike PiSSA and MiLoRA, NoRA does not depend on the spectral structure of pretrained weights, while preserving the lightweight low-rank parameterization and deployment efficiency of standard LoRA. Together with the supervised finetuning results, the RLVR experiments suggest that controlling the scale of the input-to-latent projection provides a simple and broadly effective mechanism for improving low-rank optimization.

5 Related Work and Concluding Remarks

Parameter-efficient finetuning. As LLMs continue to grow in scale and capability, there has been increasing interest in adapting them to downstream tasks in a parameter-efficient manner (Houlsby et al., 2019; Aghajanyan et al., 2020; Hu et al., 2021; Edalati et al., 2022; Wang et al., 2022; Gheini et al., 2021; Zaken et al., 2022; Guo et al., 2020; Sung et al., 2021; Ansell et al., 2022; Lester et al., 2021; Li & Liang, 2021; Vu et al., 2022; He et al., 2021; Mao et al., 2021; Karimi Mahabadi et al., 2021; Liu et al., 2022; Sung et al., 2022; Chen et al., 2023; Jia et al., 2022; Chen et al., 2022; Zhang et al., 2022; Jie & Deng, 2023; Lian et al., 2022; Qiu et al., 2023; 2025; Liu et al., 2024c; Wu et al., 2024; Zi et al., 2023). Among them, LoRA (Hu et al., 2021) is widely adopted due to its simplicity, efficiency, and mergeability at inference time. Subsequent variants improve LoRA from different perspectives, such as adaptive rank allocation (Zhang et al., 2023b), low-bit quantization (Dettmers et al., 2023) and weight-decomposed low-rank updates (DoRA) (Liu et al., 2024b). While these methods improve the capacity, efficiency, or stability of low-rank adaptation, our work focuses on a complementary question: how the low-rank subspace should be initialized for better optimization.

LoRA initialization and optimization. Recent studies show that LoRA is sensitive to initialization, scaling, and early-stage optimization dynamics. PiSSA initializes LoRA from principal singular components of pretrained weights (Meng et al., 2024), OLoRA (Büyükakyüz, 2024) adopts orthogonal initialization, LoRA-GA (Wang et al., 2024) uses gradient information to guide initialization, EVA leverages activation statistics, and rsLoRA improves stability through rank-dependent scaling (Kalajdzievski, 2023). These methods demonstrate the importance of initialization, but often rely on SVD, gradient estimation, activation statistics, or scaling heuristics. In contrast, NoRA requires no additional data, SVD, or test-time overhead while preserving the mergeability of LoRA.

Concluding remarks. This work shows that LoRA's effectiveness depends not only on its rank, but also critically on the geometry and scale of its down-projection. Inspired by the normalized latent representations in MLA, we ask whether the benefits of latent normalization can be transferred to LoRA without sacrificing its linear structure and exact mergeability. This leads to NoRA, which normalizes the down-projection along the rank dimension and thereby controls the scale of the input-to-latent mapping. From a preconditioning perspective, LoRA can be viewed as full finetuning under an implicit low-rank, input-side preconditioner determined by the down-projection; NoRA's rank-dimension normalization corrects undesirable scale imbalance in this preconditioner and yields a better-conditioned optimization geometry. NoRA can consistently improve convergence, training stability and downstream performance across pretraining, supervised finetuning, and RLVR.

AI USE STATEMENT

AI-assisted tools were used solely for language polishing, including improving grammar, clarity, and readability during manuscript preparation. The authors take full responsibility for the accuracy, originality, and integrity of the manuscript.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, Section 3 presents the mathematical formulation of NoRA and NoRA-init, including their normalization and initialization procedures. Section 4 describes the experimental frameworks, models, training datasets, and evaluation benchmarks. Appendices A, B, and C provide additional training details and key hyperparameters for pretraining, supervised finetuning, and reinforcement learning with verifiable rewards, respectively.

REFERENCES

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*, 2020.

Alan Ansell, Edoardo Ponti, Anna Korhonen, and Ivan Vulić. Composable sparse fine-tuning for cross-lingual transfer. In *ACL*, 2022.

Simran Arora, Aman Timalsina, Aaryan Singhal, Benjamin Spector, Sabri Eyuboglu, Xinyi Zhao, Ashish Rao, Atri Rudra, and Christopher Ré. Just read twice: closing the recall gap for recurrent language models. *arXiv preprint arXiv:2407.05483*, 2024.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *arXiv preprint arXiv:2505.23281*, 2025.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *AAAI*, 2020.

Kerim Büyükakyüz. Olora: Orthonormal low-rank adaptation of large language models. *arXiv preprint arXiv:2406.01775*, 2024.

Jiaao Chen, Aston Zhang, Xingjian Shi, Mu Li, Alex Smola, and Diyi Yang. Parameter-efficient fine-tuning design spaces. In *ICLR*, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. In *NeurIPS*, 2022.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In *NeurIPS*, 2023.

Ali Edalati, Marzieh Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. Krona: Parameter efficient tuning with kronecker adapter. *arXiv preprint arXiv:2212.10650*, 2022.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL `https://zenodo.org/records/12608602`.

Mozhdeh Gheini, Xiang Ren, and Jonathan May. Cross-attention is all you need: Adapting pretrained transformers for machine translation. In *EMNLP*, 2021.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Demi Guo, Alexander M Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. *arXiv preprint arXiv:2012.07463*, 2020.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*, 2024.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*, 2021.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *ICML*, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Yangyi Huang, Ruotian Peng, Zeju Qiu, Jiale Kang, Yandong Wen, Bernhard Schölkopf, and Weiyang Liu. Peft-arena: Understanding parameter-efficient finetuning from a stability-plasticity perspective. *arXiv preprint arXiv:2605.28819*, 2026.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *ECCV*, 2022.

Shibo Jie and Zhi-Hong Deng. Fact: Factor-tuning for lightweight adaptation on vision transformer. In *AAAI*, 2023.

Damjan Kalajdzievski. A rank stabilization scaling factor for fine-tuning with lora. *arXiv preprint arXiv:2312.03732*, 2023.

Jiale Kang and Qingyu Yin. Miss: Revisiting the trade-off in lora with an efficient shard-sharing structure. In *ICLR*, 2026.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. In *NeurIPS*, 2021.

Dawid Kopiczko, Tijmen Blankevoort, and Yuki Asano. Vera: Vector-based random matrix adaptation. In *ICLR*, 2024.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, 2021.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. In *NeurIPS*, 2022.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9, 2024.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, 2021.

Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Scaling & shifting your features: A new baseline for efficient model tuning. In *NeurIPS*, 2022.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*, 2024.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *NeurIPS*, 2022.

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. *arXiv preprint arXiv:2402.09353*, 2024b.

Weiyang Liu, Yan-Ming Zhang, Xingguo Li, Zhiding Yu, Bo Dai, Tuo Zhao, and Le Song. Deep hyperspherical learning. In *NeurIPS*, 2017.

Weiyang Liu, Zhen Liu, Zhiding Yu, Bo Dai, Rongmei Lin, Yisen Wang, James M Rehg, and Le Song. Decoupled networks. In *CVPR*, 2018.

Weiyang Liu, Zeju Qiu, Yao Feng, Yuliang Xiu, Yuxuan Xue, Longhui Yu, Haiwen Feng, Zhen Liu, Juyeon Heo, Songyou Peng, Yandong Wen, Michael J. Black, Adrian Weller, and Bernhard Schölkopf. Parameter-efficient orthogonal finetuning via butterfly factorization. In *ICLR*, 2024c.

Ilya Loshchilov, Cheng-Ping Hsieh, Simeng Sun, and Boris Ginsburg. ngpt: Normalized transformer with representation learning on the hypersphere. In *ICLR*, 2025.

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Wen-tau Yih, and Madian Khabsa. Unipelt: A unified framework for parameter-efficient language model tuning. *arXiv preprint arXiv:2110.07577*, 2021.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *ICML*, 2015.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. In *NeurIPS*, 2024.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

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 *EMNLP*, 2018.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. In *ACL*, 2016.

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 *NeurIPS*, 2023.

Zeju Qiu, Weiyang Liu, Adrian Weller, and Bernhard Schölkopf. Orthogonal finetuning made scalable. In *EMNLP*, 2025.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NeurIPS*, 2016.

John Schulman and Thinking Machines Lab. Lora without regret. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20250929. https://thinkingmachines.ai/blog/lora/.

Zhennan Shen, Yanshu Li, Qingyu Yin, Chak Tou Leong, Zhilin Wang, Yanxu Chen, Rongduo Han, Sunbowen Lee, and Yi R Fung. On the geometry of on-policy distillation. *arXiv preprint arXiv:2606.07082*, 2026.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama, 2023.

Yi-Lin Sung, Varun Nair, and Colin A Raffel. Training neural networks with fixed sparse masks. *NeurIPS*, 2021.

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. In *NeurIPS*, 2022.

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.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. Spot: Better frozen model adaptation through soft prompt transfer. In *ACL*, 2022.

Hanqing Wang, Yixia Li, Shuo Wang, Guanhua Chen, and Yun Chen. Milora: Harnessing minor singular components for parameter-efficient llm finetuning. In *NAACL-HLT*, 2025.

Shangshang Wang, Julian Asilis, Ömer Faruk Akgül, Enes Bilgin, Ollie Liu, and Willie Neiswanger. Tina: Tiny reasoning models via lora. In *ICLR*, 2026.

Shaowen Wang, Linxi Yu, and Jian Li. Lora-ga: Low-rank adaptation with gradient approximation. In *NeurIPS*, 2024.

Yaqing Wang, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. Adamix: Mixture-of-adapter for parameter-efficient tuning of large language models. In *EMNLP*, 2022.

Taiqiang Wu, Jiahao Wang, Zhe Zhao, and Ngai Wong. Mixture-of-subspaces in low-rank adaptation. *arXiv preprint arXiv:2406.11909*, 2024.

Songlin Yang and Yu Zhang. Fla: A triton-based library for hardware-efficient implementations of linear attention mechanism, January 2024. URL `https://github.com/fla-org/flash-linear-attention`.

Muchao Ye, Weiyang Liu, and Pan He. Vera: Explainable video anomaly detection via verbalized learning of vision-language models. In *CVPR*, 2025.

Qingyu Yin, Yulun Wu, Zhennan Shen, Sunbowen Li, Zhilin Wang, Yanshu Li, Chak Tou Leong, Jiale Kang, and Jinjin Gu. Evaluating parameter efficient methods for rlvr. *arXiv preprint arXiv:2512.23165*, 2025.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. In *ICLR*, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. In *NeurIPS*, 2026.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models. In *ACL*, 2022.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *ACL*, 2019.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. Lora-fa: Efficient and effective low rank representation fine-tuning. *arXiv preprint arXiv:2308.03303*, 2023a.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023b.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Yuanhan Zhang, Kaiyang Zhou, and Ziwei Liu. Neural prompt search. *arXiv preprint arXiv:2206.04673*, 2022.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. Opencodeinterpreter: Integrating code generation with execution and refinement. In *Findings of ACL*, 2024.

Bojia Zi, Xianbiao Qi, Lingzhi Wang, Jianan Wang, Kam-Fai Wong, and Lei Zhang. Delta-lora: Fine-tuning high-rank parameters with the delta of low-rank matrices. *arXiv preprint arXiv:2309.02411*, 2023.

A PRETRAINING SETTINGS

We conduct the pre-training experiments on FineWeb-10BT. Unless otherwise specified, all models are trained for 20,480 optimization steps with a sequence length of 2,048 and a global batch size of 256. We use AdamW with a peak learning rate of 3×10^{-4} and $\epsilon = 10^{-15}$. The learning rate is warmed up for the first 1,024 steps and then decayed using a cosine schedule to 10% of the peak learning rate. The gradient norm is clipped at 1.0. All experiments use the same training configuration and random seed for controlled comparison.

Hyperparameter	**Value**
Dataset	FineWeb-10BT
Optimizer	AdamW
Learning rate	3×10^{-4}
Adam ϵ	10^{-15}
LR scheduler	Cosine decay
Warmup steps	1,024
Minimum LR ratio	0.1
Global batch size	256
Sequence length	2,048
Training steps	20,480
Gradient clipping	1.0
Random seed	42

Table 7: Hyperparameter settings for the pretraining experiments.

B SUPERVISED FINETUNING SETTINGS

For the SFT experiments, we tune the hyperparameters of each method individually to ensure competitive performance. For NoRA, we recommend using $\alpha = r$ as the default setting, under which the early gradient norm is close to that of full finetuning. Although a larger scaling factor, such as $\alpha = 2r$, can yield better loss fitting in some cases, we observe that it may also lead to increased forgetting of the pretrained knowledge. Considering this trade-off between adaptation and retention, we use and recommend $\alpha : r = 1 : 1$ as the default configuration.

Hyperparameters	**LoRA**	**DoRA**	**PiSSA**	**rsLoRA**	**OFT**	**NoRA**
rank	32	32	32	32	32	32
α	64	64	32	64	-	32
dropout	0.0					
optimizer	AdamW					
lr	2e-5					
lr scheduler	Cosine decay					
batch size	128					
warmup ratio	0.3					
epochs	1					
target_modules	Q,K,V,O,Up,Down,Gate					

Table 8: Hyperparameter settings for the Llama-3.2-3B supervised finetuning experiments.

C Reinforcement Learning Settings

We conduct reinforcement learning experiments on DeepSeek-R1-Distill-Qwen-1.5B using the DAPO objective and the DAPO-Math-17K dataset. All experiments are performed with 8 GPUs using bfloat16 precision. We train for 1,024 optimization steps with a global batch size of 128 and a learning rate of 1×10^{-5} under a cosine learning-rate schedule without warmup. For each prompt, we sample 8 responses with a maximum completion length of 16,384 tokens. The maximum prompt length is set to 512 tokens.

For parameter-efficient training, we use a rank of 32 and set $\alpha = 64$. The low-rank modules are applied to all attention projections (q, k, v, and o) and MLP projections (up, down, and gate). We use the same training configuration across different methods to ensure a controlled comparison.

Hyperparameter	**Value**
Model	DeepSeek-R1-Distill-Qwen-1.5B
Dataset	DAPO-Math-17K
RL objective	DAPO
Precision	bfloat16
Learning rate	1×10^{-5}
LR scheduler	Cosine decay
Warmup ratio	0
Global batch size	128
Training steps	1,024
LoRA rank r	32
LoRA α	64
LoRA dropout	0.05
Responses per prompt	8
Maximum prompt length	512
Maximum completion length	16,384
ϵ_{high}	0.28
β	0.0
Random seed	42

Table 9: Hyperparameter settings for the reinforcement learning experiments.