RanLoRA: Residual-aware Nonlinear Low-Rank Adaptation

Xu Luo, Yongbin Liu*, Chunping Ouyang, Ying Yu
School of Computer, University of South China
{xuluo1010, yongbinliu03}@gmail.com,
{ouyangcp}@126.com, {yuying}@usc.edu.cn

Abstract

Low-Rank Adaptation (LoRA) is a widely adopted approach for parameter-efficient fine-tuning of large language models, enabling effective adaptation with a small number of trainable parameters. However, its reliance on linear low-rank projections restricts adaptation to linear subspaces, which can limit flexibility on complex downstream tasks. To address this, we propose **RanLoRA**, a **R**esidual-**a**ware **N**onlinear **Lo**w-**R**ank **A**daptation approach that leverages the decomposition structure of pretrained weights. We used Singular Value Decomposition (SVD) to decompose pretrained weights into principal components that are kept frozen and residual components that are used for task-specific adaptation. To enhance the expressiveness of linear low-rank updates, RanLoRA incorporates a nonlinear activation layer together with a Hadamard-product-based vector modulation. This design supports an implicit progressive adaptation behavior, where optimization evolves from coarse approximation of dominant components toward residual alignment and fine-grained nonlinear refinement. Experiments on benchmarks covering commonsense reasoning, natural language understanding, image classification, and mathematical reasoning show that RanLoRA consistently outperforms vanilla LoRA and representative variants under comparable parameter budgets. These results suggest that incorporating structured nonlinearity into adapter design can enhance representational flexibility and generalization across tasks in large models.

1 Introduction

Pre-trained models have become the cornerstone of modern machine learning systems, exhibiting strong generalization ability by leveraging large-scale and diverse training corpora. Their effectiveness has been demonstrated across a broad spectrum of tasks, including natural language understanding (Devlin et al., 2019; Liu et al., 2019), natural language generation (Touvron et al., 2023; AI, 2024), and vision applications such as image classification (Dosovitskiy et al., 2021). Despite their success, adapting these models to downstream tasks typically relies on full fine-tuning, which becomes increasingly impractical as model sizes grow, due to substantial computational and memory requirements (Qin et al., 2024). This challenge has driven extensive research into parameter-efficient fine-tuning (PEFT) methods (Ding et al., 2023; Han et al., 2024), which aim to retain adaptability while significantly reducing resource consumption by introducing lightweight trainable components (Lin et al., 2024).

Model	Method	Param	Mem	Time	Avg.
LLaMA3-8B	LoRA	3.5M	29.6G	6.6h	83.6
	MoSLoRA	3.5M	30.4G	7.6h	83.6
	NEAT	3.5M	30.6G	7.2h	83.7
	AuroRA	3.5M	30.8G	11.8h	83.9
	RanLoRA	3.5M	30.2G	6.9h	**87.2**

Table 1: **Param** denotes the number of trained parameters, **Time** is the time required for training and testing on H800 GPU, **Mem** for the GPU Memory usage, and **Avg.** means the average accuracy on commonsense reasoning tasks. To ensure a fair comparison, we conduct all tests within the same code framework LLM-Adapters (Hu et al., 2023), the details of hyperparameters settings are provided in Appendix C.1.

Among existing PEFT methods, Low-Rank Adaptation (LoRA) and its variants (Hu et al., 2022; Liu et al., 2024; Song et al., 2024; Büyükakyüz, 2024; Zhao et al., 2024) stand out for their simplicity and effectiveness. Rather than updating full weight matrices, LoRA parameterizes task-specific updates through the product of two low-rank matrices, thereby reducing both the number of trainable parameters and the memory footprint during fine-tuning. This formulation has proven effective across a wide range of scenarios, contributing to LoRA's widespread adoption. However, the effi-

*Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 17243–17258
July 2-7, 2026 ©2026 Association for Computational Linguistics

ciency of LoRA comes with an inherent trade-off. Restricting weight updates to a low-rank linear subspace limits the expressive capacity of the adaptation, particularly when modeling complex task-specific transformations (Pan et al., 2024). In practice, performance often degrades noticeably in low-rank regimes, as the constrained parameterization fails to capture intricate optimization trajectories. While increasing the rank can partially alleviate this issue, it simultaneously leads to higher parameter overhead, diminishing the parameter-efficiency advantages that motivate LoRA in the first place.

To overcome these limitations, a number of extensions to vanilla LoRA have been proposed. Approaches such as MoSLoRA (Wu et al., 2025) introduce mixer matrices to blend multiple subspaces, whereas Pissa (Meng et al., 2024) and MiLoRA (Wang et al., 2024) exploit SVD-based initialization to improve convergence behavior. Beyond linear adaptations, NEAT (Zhong et al., 2025) incorporates a lightweight neural network to accumulate weight updates, substantially enhancing expressiveness at the cost of increased architectural complexity. More recently, AuroRA (Dong et al., 2025) further extends LoRA by inserting an Adaptive Nonlinear Layer (ANL) between linear projections, forming an MLP-like adaptation structure. Despite these advances, an important limitation remains. Existing nonlinear approaches, including NEAT and AuroRA, primarily treat the adaptation module as a unified black-box transformation, without explicitly considering the decomposition of pre-trained weights into principal and residual components. In particular, nonlinear transformations are applied uniformly across all components, failing to distinguish dominant principal components, which are often associated with robust and generalizable knowledge, from residual components that encode finer-grained, task-specific variations. This lack of explicit principal–residual separation can lead to inefficient parameter allocation and suboptimal optimization dynamics, especially under tight parameter budgets.

To address this issue, we propose RanLoRA, a nonlinear adaptation approach grounded in explicit principal–residual decoupling. In contrast to generic MLP-style nonlinear adapters such as AuroRA, RanLoRA leverages Singular Value Decomposition (SVD) to explicitly separate weight updates into a principal linear path and a residual nonlinear path. Nonlinear adaptation is then applied exclusively to the residual components through a lightweight vector-stretching modulation implemented via element-wise scaling, while the principal components remain unchanged. This design preserves core pretrained knowledge and enables a progressive fitting process that refines task-specific directions in a parameter-efficient manner.

As shown in Table 1, RanLoRA achieves a favorable balance between effectiveness and efficiency, delivering consistent performance improvements over prior PEFT methods with comparable, while maintaining moderate training time and memory usage.

Our contributions can be summarized as follows:

- We present RanLoRA, a residual-aware nonlinear LoRA variant that systematically combines SVD-based residual-aware initialization with lightweight nonlinear modulation. By separating principal and residual components of pretrained weights, RanLoRA enables targeted nonlinear refinement while preserving pretrained knowledge.
- We introduce a progressive fitting strategy that evolves from a coarse linear approximation to fine-grained nonlinear adaptation through lightweight vector-stretching modulation, significantly enhancing expressiveness without increasing architectural complexity.
- Extensive experiments across reasoning, understanding, and computer vision benchmarks demonstrate that RanLoRA consistently improves performance under constrained parameter budgets, offering a strong trade-off between accuracy and parameter efficiency.

2 Related Works

2.1 Low-Rank Adaptation

Parameter-efficient fine-tuning (PEFT) minimizes the high costs of updating large language models (LLMs) by optimizing only a small parameter subset. Low-Rank adaptation, pioneered by LoRA (Hu et al., 2022), factorizes weight updates into two low-rank matrices, ensuring minimal parameters and zero inference overhead. Recent variants enhance this strategy: DoRA (Liu et al., 2024) utilizes magnitude-direction decomposition, while MoRA (Jiang et al., 2024) employs input compression for higher-rank updates. Others leverage Fourier transforms (FourierFT (Gao et al., 2024a)), singular component updates (PiSSA (Meng et al.,

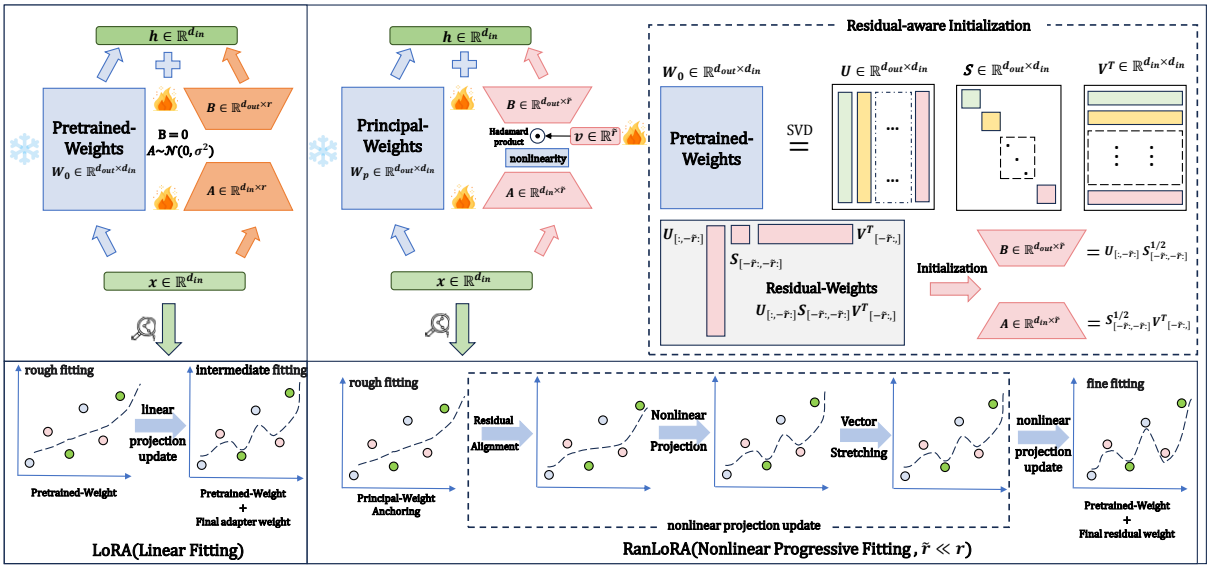

Figure 1: Comparison of architecture and fitting strategy. (Left) **LoRA**: A two-layer linear projection with hidden dimension r, where adaptation is confined to a fixed linear subspace. (Right) **RanLoRA (Ours)**: Nonlinear projection enabled by a compact latent bottleneck ($\hat{r} \ll r$).As illustrated by the refinement curves, RanLoRA follows a progressive fitting process that evolves from principal weight anchoring to vector-stretching modulation, allowing accurate task-space approximation with a minimal rank.

2024), MiLoRA (Wang et al., 2024)), or nonlinear weight modeling (NEAT (Zhong et al., 2025), AuroRA (Dong et al., 2025)). Together, these methods strive to balance representational expressiveness with computational efficiency.

2.2 Other PEFT Methods

Beyond low-rank techniques, prompt-based and adapter-based strategies are prominent. Prompt-based methods, such as Prompt Tuning (Lester et al., 2021) and P-Tuning (Liu et al., 2022), inject trainable virtual tokens into input embeddings. Despite their minimal footprint, they are sensitive to initialization (Wu et al., 2024) and increase inference costs due to the Transformer's quadratic complexity (Vaswani et al., 2017). Adapter-based methods, including original Adapters (Houlsby et al., 2019), Compacter (Karimi Mahabadi et al., 2021), and Parallel Adapters (He et al., 2021), insert lightweight modules into the pre-trained backbone. While offering excellent modularity and performance, they inherently alter the model architecture, introducing structural overhead that low-rank approaches typically avoid.

3 Methodology

3.1 Preliminary

LoRA((Hu et al., 2022)) assumes that weight updates during fine-tuning follow a low-rank structure. Specifically, the update of a pre-trained weight matrix $W_0 \in \mathbb{R}^{d_{out} \times d_{in}}$ is approximated by the product of two learnable low-rank matrices:

$$W = W_0 + \Delta W = W_0 + BA, \tag{1}$$

where $B \in \mathbb{R}^{d_{out} \times r}$, $A \in \mathbb{R}^{r \times d_{in}}$, and $r \ll \min(d_{out}, d_{in})$. During fine-tuning, the original weights W_0 remain frozen, while only the introduced matrices B and A are optimized by solving

$$\min_{B,A} L(D_{\text{train}}; W_0 + BA), \tag{2}$$

with D_{train} denoting the training dataset and L is the loss function. Since both B and A are low-rank matrices with far fewer parameters than W_0, LoRA significantly reduces memory and computation overhead compared to full fine-tuning when adapted to downstream tasks (Long et al., 2024).

3.2 Residual-aware Initialization

Given a pretrained weight matrix $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, we explicitly decompose it into a principal component and a residual component to enable residual-aware adaptation. Using Singular Value Decomposition (SVD), we express

$$W_0 = USV^\top, \tag{3}$$

where the matrices $U \in \mathbb{R}^{d_{out} \times d_{out}}$, $V^\top \in \mathbb{R}^{d_{in} \times d_{in}}$ are orthogonal matrices containing the

left and right singular vectors, and $V^\top$ is the transpose of V. $S \in \mathbb{R}^{d_{out} \times d_{in}}$ is a non-negative diagonal matrix whose diagonal elements are the singular values arranged in descending order.

We divide them into two complementary parts: the principal component of pretrained weights

$$W_P = U_{[:,:m]} S_{[:m,:n]} V^\top_{[:n,:]}, \tag{4}$$

where $m = d_{out} - \hat{r} - 1$ and $n = d_{in} - \hat{r} - 1$, capturing dominant pretrained directions, and the rest residual component of pretrained weights

$$W_R = U_{[:,-\hat{r}:]} S_{[-\hat{r}:,-\hat{r}:]} V^\top_{[-\hat{r}:,:]}, \tag{5}$$

$\hat{r} \ll r$, representing lower-energy directions associated with finer-grained variations. Formally,

$$W_0 = W_P + W_R. \tag{6}$$

The residual components of weights are used to initialize the matrices A and B in RanLoRA:

$$B = U_{[:,-\hat{r}:]} S^{1/2}_{[-\hat{r}:,-\hat{r}:]}, A = S^{1/2}_{[-\hat{r}:,-\hat{r}:]} V^\top_{[-\hat{r}:,:]}. \tag{7}$$

The principal components of pretrained weights remain frozen during fine-tuning, which ensures that nonlinear adaptation does not distort dominant principal components.

3.3 The RanLoRA Architecture

Building upon the residual-aware initialization, the modified model's forward propagation is formulated as:

$$h = W_p x + B\left(v \odot GELU(Ax)\right) \tag{8}$$

where $x \in \mathbb{R}^{d_{\text{in}}}$ is the input hidden state, $A \in \mathbb{R}^{\hat{r} \times d_{\text{in}}}$ and $B \in \mathbb{R}^{d_{\text{out}} \times \hat{r}}$ are low-rank matrices with $\hat{r} \ll r$. We used GELU due to its innate gating mechanism, and v is a learnable modulation vector.

In this formulation, the nonlinear transformation operates exclusively on residual directions. The vector v performs an element-wise Hadamard product ($\odot$) in the bottleneck, acting as a dynamic scaling factor that stretches or compresses the feature importance across different latent dimensions.

3.4 Progressive Fitting Strategy

The proposed residual-aware nonlinear adaptation naturally induces a progressive fitting behavior, where optimization evolves from stable principal anchoring to fine-grained residual refinement. As illustrated in Figure 1, the adaptation unfolds through four hierarchical stages. We note that these stages do not correspond to explicitly separated optimization phases, but serve as an interpretive framework for understanding how nonlinear residual adaptation emerges during training:

Principal Weight Anchoring: We anchor the principal components of pretrained weights W_p by freezing it to maintain the stability of the backbone.

Residual Alignment: The coupled matrices (A, B) are aligned with the residual subspace, identifying specific directions of variance that require task-specific refinement.

Nonlinear Projection: The residual flow is passed through a nonlinear activation function to capture high-order feature dynamics and break the flat subspace constraint inherent in vanilla LoRA.

Fine-grained Vector Stretching: the modulation vector v performs precise adjustment within the residual unit. This enables RanLoRA to achieve a "fine fitting" of the task distribution, accurately capturing complex optimization trajectories with minimal parameter overhead.

3.5 Complexity Analysis

Parameter Efficiency. In vanilla LoRA, the number of trainable parameters scales as $O(r(d_{in} + d_{out}))$, where r is the adaptation rank. For RanLoRA, we significantly compress the parameter footprint by employing a minimal latent rank $\hat{r} \ll r$ (e.g., $\hat{r} = 2$). The introduction of the modulation vector $v \in \mathbb{R}^{\hat{r}}$ adds only $\hat{r}$ additional parameters. Consequently, the total parameter count for RanLoRA is $O(\hat{r}(d_{in} + d_{out}) + \hat{r})$. Given that $\hat{r}$ is multiple times smaller than the r used in linear LoRA, RanLoRA achieves a substantial reduction in memory requirements while delivering superior expressiveness through its nonlinear residual path. As shown in Table 1, RanLoRA achieves a favorable balance between effectiveness and efficiency which is closed to vanilla LoRA due to the minimal computational overhead from rank setting is 2. The total time for performing SVD decomposition on the weights is 541.49 seconds (≈0.15 h) during the training initialization. We emphasize that SVD is a one-time pre-computation step performed only at the very beginning of fine-tuning. For the entire fine-tuning process, which typically spans several hours or days, this 0.15-hour overhead is negligible.

Computational Complexity. The forward pass of RanLoRA, defined as $\Delta h = B(v \odot GELU(Ax))$, involves three primary operations.

Let b denote the batch size. (1) The linear projection Ax incurs a complexity of $O(bd_{in}\hat{r})$. (2) The latent operations, comprising the GELU and the Hadamard product with v, each contribute $O(b\hat{r})$. (3) The final projection $B(\cdot)$ incurs $O(bd_{out}\hat{r})$. The total complexity is thus $O(b\hat{r}(d_{in}+d_{out}+1)+b\hat{r})$. Under the standard low-rank regime where $\hat{r} \ll \min(d_{in}, d_{out})$, the overhead introduced by GELU and v is linear with respect to the latent dimension and mathematically negligible compared to the dominant projection terms. Thus, RanLoRA maintains a computational footprint comparable to vanilla LoRA, despite its enhanced nonlinear modeling capability.

4 Experiment

In this section, we conduct experiments on four tasks to evaluate the proposed RanLoRA method.

4.1 Datasets

We experiment RanLoRA on datasets from four representative benchmarks.

4.1.1 Commonsense Reasoning

We conduct experiments on Commonsense170K, a benchmark proposed by (Hu et al., 2023) that unifies eight commonsense reasoning sub-tasks with standardized training and testing splits. This collection covers diverse aspects of commonsense reasoning, including BoolQ (Clark et al., 2019), PIQA(Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2020), ARC-e and ARC-c (Clark et al., 2018), and OBQA (Mihaylov et al., 2018).

4.1.2 Natural Language Understanding

Natural Language Understanding (NLU) experiments are conducted on eight datasets from the GLUE benchmark (Wang et al., 2018), which cover a wide spectrum of linguistic phenomena including entailment, paraphrase detection, sentiment analysis, and sentence similarity. Specifically, GLUE comprises tasks such as QNLI, RTE, SST-2, MRPC, CoLA and STS-B. To ensure comparability with prior work, we adopt the evaluation metrics and experimental protocols established in (Gao et al., 2024a), where Matthew's correlation for CoLA, Pearson/Spearman correlations for STS-B and accuracy is used for the other tasks.

4.1.3 Image Classification

Image Classification is evaluated on eight benchmarks following (Gao et al., 2024b): Oxford-Pets (Parkhi et al., 2012), CIFAR10 (Krizhevsky, 2009), DTD (Cimpoi et al., 2014), EuroSAT (Helber et al., 2019), RESISC45 (Cheng et al., 2017), StanfordCars (Krause et al., 2013), FGVC (Maji et al., 2013), and CIFAR100 (Krizhevsky, 2009). The first five datasets represent small label spaces (10–47 categories), while the latter three involve large, fine-grained label spaces (100+ categories). This setup enables evaluation across both general and fine-grained classification scenarios.

4.1.4 Mathematical Reasoning

For this task, we employ MetaMath (Yu et al., 2023) as the training corpus and GSM8K (Cobbe et al., 2021) as the test dataset. Models need to generate correct answers, and accuracy is used as the evaluation metric.

Further details regarding the baselines, datasets and hyperparameters are provided in Appendix A, B, C.

4.2 Results on Commonsense Reasoning Tasks

Table 2 reports the performance of RanLoRA compared to existing PEFT methods across eight benchmarks using LLaMA3-8B. Several key observations emerge:

Best Performance with Minimal Parameters. RanLoRA achieves an average accuracy of 87.2%, outperforming all baselines, including recent nonlinear and mixture-of-subspaces methods. Compared to the previous best-performing model MoSLoRA (84.9%), RanLoRA attains a +2.3% absolute improvement while utilizing only 3.5M trainable parameters, which corresponds to approximately 1/8 of MoSLoRA's and 1/16 of LoRA's parameter footprint.

Superiority over Nonlinear Baselines. Under the same 3.5M parameter budget, RanLoRA surpasses AuroRA by +3.3% on average. Although AuroRA performs competitively on HellaSwag, it exhibits lower performance on complex reasoning benchmarks such as ARC-c and BoolQ. In contrast, RanLoRA consistently outperforms all baselines, particularly on ARC-e (+3.7% over AuroRA) and OBQA (+4.0%). These results indicate that the nonlinear adaptation scheme of RanLoRA more effectively captures task-specific information than generic MLP-style layers.

Breaking the Linear Bottleneck. Linear LoRA variants such as PiSSA and MiLoRA fail to match RanLoRA despite substantially larger parameter counts. For instance, PiSSA (83.8M parameters) is 12.4% lower on average than RanLoRA. This

Model	Method	Params	ARC-e	OBQA	SIQA	ARC-c	WinoG	PIQA	BoolQ	HellaS	Avg.
LLaMA3-8B	LoRA*	56.6M	84.2	79.0	79.9	71.2	84.3	85.2	70.8	91.7	80.8
	PiSSA*	83.8M	77.7	74.6	77.2	63.2	78.9	81.1	67.1	83.6	75.4
	MiLoRA*	56.6M	86.8	81.9	77.2	75.5	85.6	86.7	68.8	92.9	81.9
	MoSLoRA	28.3M	90.1	84.2	79.8	79.8	86.6	88.6	74.6	95.1	84.9
	NEAT	56.6M	91.3	84.2	80.9	78.2	86.8	80.9	72.7	94.3	83.7
	AuroRA	3.5M	89.4	84.8	79.7	78.3	83.9	87.1	73.5	94.3	83.9
	RanLoRA	3.5M	**93.1**	**88.8**	**83.5**	**83.1**	**88.5**	**89.8**	**75.1**	**95.8**	**87.2**

Table 2: Commonsense reasoning performance on LLaMA3-8B. Results marked with "*" are taken from (Wang et al., 2024). Best results are highlighted in **bold**. "Avg." means the average accuracy of all datasets.The experiments are repeated 5 times under random seeds 42 to 46 and the average performance is reported.

demonstrates that simply increasing the rank r in a linear framework yields diminishing returns compared to the progressive fitting strategy.

4.3 Results on NLU Tasks

We evaluate RanLoRA on the GLUE benchmark with RoBERTa-base and RoBERTa-large backbones. As shown in Table 3, RanLoRA consistently outperforms competitive baselines across different model scales.

Setting New Performance Upper Bounds. On RoBERTa-base, RanLoRA achieves an average score of 86.7%, exceeding Full Fine-Tuning (FFT, 85.2%) and the linear variant DoRA (85.5%). This performance is attained with only 0.074M trainable parameters, corresponding to 1/4 of DoRA's budget and 1/1700 of FFT, demonstrating that the nonlinear residual mapping effectively refines linguistic representations in a compact parameter space.

Superior Performance on Complex Linguistic Tasks. RanLoRA shows notable gains on tasks requiring deep linguistic understanding, such as CoLA (+2.6% over DoRA) and QNLI (+0.7% over DoRA). On RoBERTa-large, RanLoRA achieves 69.8% on CoLA, surpassing the nonlinear method AuroRA (68.5%). These results suggest that RanLoRA better captures linguistic complexities, including syntax and logical entailment, than generic nonlinear or frequency-domain approaches.

Scalability and Consistency. When the model is scaled to RoBERTa-large, RanLoRA attains an average score of 89.4%, maintaining a lead of +1.2% over FFT and +0.6% over AuroRA. Consistent improvements across all six sub-tasks indicate the robustness of the nonlinear adaptation architecture, confirming RanLoRA as an effective and general-purpose framework for large-scale language models.

4.4 Results on Image Classification Tasks

As shown in Table 4, the robustness of RanLoRA in the computer vision domain is evaluated on image classification tasks using ViT-base and Vit-large models across eight diverse datasets. The main observations are summarized as follows:

Exceptional Parameter Efficiency at Base Scale. On ViT-Base, RanLoRA uses only 73K trainable parameters, corresponding to 12.5% of LoRA's budget, while achieving a +8.1% improvement in average accuracy. In particular, our method RanLoRA (85.7%) surpasses the competitive baseline NEAT (85.3%) with only 27.8% of its trainable parameters, indicating that the nonlinear adaptation strategy is more resource-efficient than auxiliary network-based approaches.

Scaling Advantages on ViT-Large. The benefits of RanLoRA are amplified as the backbone scale increases. On ViT-Large, RanLoRA attains an average accuracy of 88.6%, outperforming AuroRA by +1.9% while requiring fewer parameters. Notably, it achieves superior performance on challenging fine-grained tasks, including FGVC (+3.5% over AuroRA) and Cars (+1.2% over AuroRA), demonstrating its effectiveness in handling the complex, high-dimensional feature spaces of large-scale vision models.

4.5 Results on Mathematical Reasoning Task

The performance of RanLoRA is evaluated on mathematical reasoning tasks using the MetaMath dataset for training and GSM8K for evaluation. Table 5 compares RanLoRA with competitive baselines. We could observe below observations:

Surpassing Full Fine-Tuning with Minimal Parameters. RanLoRA ($r = 8$, 14M parameters) achieves an accuracy of 66.7%, establishing a new state-of-the-art among PEFT methods. It also surpasses Full Fine-Tuning (66.5%) while utilizing only 0.2% of its trainable parameters, demonstrating that RanLoRA can optimize task-specific logi-

Model	Method	Param	SST-2	MRPC	CoLA	QNLI	RTE	STS-B	Avg.
RoBERTa-Base	FFT*	125M	94.8	90.2	63.6	92.8	78.7	91.2	85.2
	BitFit*	0.1M	93.7	**92.7**	62.0	91.8	81.5	90.8	85.4
	Adapter^{D}*	0.9M	94.7	88.4	62.6	93.0	75.9	90.3	84.2
	LoRA*	0.3M	95.1	89.7	63.4	93.3	78.4	91.5	85.2
	AdaLoRA*	0.3M	94.5	88.7	62.0	93.1	81.0	90.5	85.0
	DyLoRA*	0.3M	94.3	89.5	61.1	92.2	79.1	90.8	85.0
	FourierFT	0.024M	94.2	90.0	63.8	92.2	79.1	90.8	85.0
	LoRA-drop*	0.15M	94.5	89.5	62.9	93.1	81.4	91.0	85.4
	DoRA*	0.3M	95	89.7	64.9	92.9	79.2	91.3	85.5
	NEAT	0.3M	94.5	88.2	64.6	92.8	78.0	91.0	84.9
	AuroRA	0.075M	95.2	90.9	64.5	92.6	**82.4**	90.5	86.0
	RanLoRA	0.074M	**95.7**	91.4	**67.5**	**93.6**	80.5	**91.6**	**86.7**
RoBERTa-Large	FFT*	356M	96.4	90.9	68.0	94.7	86.6	92.4	88.2
	Adapter^{P}*	3M	96.1	90.2	68.3	94.8	83.8	92.1	87.6
	Adapter^{H}*	6M	96.2	88.7	66.5	94.7	83.4	91.0	86.8
	LoRA*	0.8M	96.2	90.2	68.2	94.8	85.2	92.3	87.8
	AuroRA	0.2M	95.8	91.3	68.5	95.0	**89.9**	92.5	88.8
	RanLoRA	0.2M	**96.9**	**91.2**	**69.8**	**95.7**	89.2	**93.3**	**89.4**

Table 3: We report the performance of different fine-tuning methods on six datasets of the GLUE benchmark, using RoBERTa-Base and RoBERTa-Large models. For CoLA, we report the Matthew's Correlation Coefficient (MCC); for STS-B, we report the Pearson Correlation Coefficient (PCC); and for all other tasks, we report accuracy (Acc.). The reported results are the average of five runs, each using a different random seed. Results marked with "*" are taken from (Dong et al., 2025). The best results are highlighted in **bold** (excluding FFT).

Model	Method	Param	OxfordPets	Cars	CIFAR10	DTD	EuroSAT	FGVC	RESISC45	CIFAR100	Avg.
Vit-Base	FFT*	85.8M	93.1	79.8	98.9	77.7	99.1	54.8	96.1	92.4	86.5
	LP*	-	90.3	25.8	96.4	69.8	88.7	17.4	74.2	84.3	68.4
	LoRA*	581K	93.2	45.4	98.8	75.0	98.4	25.2	92.7	92.0	77.6
	FourierFT*	239K	93.1	56.4	98.7	77.3	**98.8**	32.4	**94.3**	91.5	80.3
	NEAT	263K	92.6	**77.3**	98.7	78.7	98.4	**51.3**	93.7	92.0	85.3
	AuroRA	74K	93.7	75.9	98.0	76.5	98.7	48.3	93.5	92.2	84.6
	RanLoRA	73K	**94.3**	76.5	**98.9**	**80.5**	**98.8**	50.5	94.0	**92.4**	**85.7**
Vit-Large	FFT*	303.3M	94.4	88.9	99.2	81.8	99.0	68.3	96.4	93.6	90.2
	LP*	-	91.1	37.9	97.8	73.3	92.6	24.6	82.0	84.3	73.0
	LoRA*	1.57M	94.8	73.3	**99.1**	81.8	98.6	42.3	94.7	**94.9**	84.9
	FourierFT*	480K	94.8	79.1	**99.1**	81.9	98.7	51.3	95.2	93.4	86.7
	AuroRA	197K	94.4	82.0	95.8	78.3	**98.8**	56.7	94.4	92.9	86.7
	RanLoRA	195K	**95.5**	**83.2**	**99.1**	**82.5**	98.9	**60.2**	**95.8**	93.3	**88.6**

Table 4: Fine-tuning results with ViT-Base and Vit-Large models on different image classification datasets. We report the accuracy (%) after 10 epochs. Results marked with "*" are taken from (Gao et al., 2024b). The best results are highlighted in **bold** (excluding FFT).

cal reasoning more effectively than global weight updates.

Model	Method	Param	GSM8K
LLaMA2-7B	FFT*	6.7B	66.5
	LoRA*$_{r=64}$	113.2M	60.6
	PiSSA*$_{r=64}$	167.6M	58.2
	MiLoRA*$_{r=64}$	113.2M	63.5
	NEAT*$_{r=64}$	113.2M	65.2
	AuroRA$_{r=8}$	14.1M	59.9
	RanLoRA$_{r=8}$	14.1M	**66.7**

Table 5: Results on mathematical reasoning task. The best result is highlighted in **bold**. Results marked with "*" are taken from (Zhong et al., 2025).

Efficiency Across Diverse Baselines. Compared to NEAT (65.2%) and MiLoRA (63.5%), which employ significantly higher ranks ($r = 64$, 113.2M parameters), RanLoRA ($r = 8$) achieves superior performance at 1/8 of the parameter cost. Furthermore, at the same rank ($r = 8$), RanLoRA (66.7%) outperforms AuroRA (59.9%) by +6.8%. These results indicate that the nonlinear residual path in RanLoRA provides a more effective representation for mathematical reasoning than merely increasing the dimensionality in linear frameworks.

4.6 Study

4.6.1 Ablation Study

An ablation study is conducted on four commonsense reasoning datasets to evaluate the contribution of each component in RanLoRA (Table 6).

Effect of Residual-aware Initialization. Replacing the proposed residual-aware initialization

scheme with standard Kaiming initialization (*w/o rai*) results in a consistent performance decrease. This indicates that anchoring the adaptation matrices to the residual singular space of W_0 provides a more effective inductive bias than random initialization, facilitating the transfer of pre-trained knowledge to downstream tasks.

Effect of Nonlinearity. Removing the nonlinear activation function (*w/o act*), leading to a performance drop (e.g., -1.2% on BoolQ). This demonstrates that nonlinear projection is critical for escaping the subspace limitations of vanilla LoRA and for capturing complex feature dynamics within the latent bottleneck.

Effect of the Modulation Vector. The most pronounced degradation occurs when the modulation vector v is removed (*w/o v*), with accuracy decreasing by 1.9% on ARC-c and 2.0% on BoolQ. These results indicate that v is the key driver of fine-grained adaptation. By modulating the nonlinear manifold, v enables precise alignment with the task distribution; removing v collapses RanLoRA to a standard nonlinear bottleneck, confirming that dimension-wise modulation is critical.

Settings	OBQA	SIQA	ARC-c	BoolQ
RanLoRA	88.8	83.5	83.1	75.1
RanLoRA w/o rai	88	82	82.5	74.7
RanLoRA w/o act	86.8	82.1	82.3	73.9
RanLoRA w/o v	86.8	81.7	81.2	73.1

Table 6: Comparison of different component on four commonsense reasoning datasets.

4.6.2 Effect of Activation Function

The impact of different activation functions on RanLoRA was evaluated across four commonsense reasoning datasets. Results are summarized in Table 7, showing that RanLoRA is compatible with multiple nonlinearities, with certain functions yielding superior performance.

Peak performance is achieved when using GELU activation function. In particular, GELU outperforms the traditional ReLU by +1.2% on OBQA and +1.6% on SIQA. This improvement suggests that the smooth, innate gating mechanism properties of GELU are better suited for mapping complex task manifolds in the latent residual space. While ReLU maintains competitive results on datasets such as HellaSwag, Tanh shows a noticeable decline, especially on WinoGrande. These observations indicate that the choice of activation functions is a significant factor in the fine-fitting stage of RanLoRA, with smoother, gating-like activations facilitating more robust gradient flow for latent vector modulation.

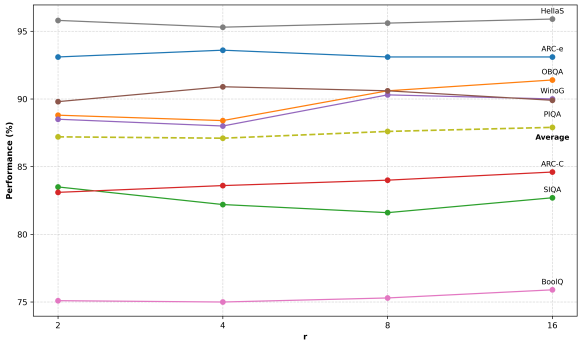

Figure 2: Performance comparison of RanLoRA with different ranks. We finetune the pretrained LLaMA3-8B model on datasets for commonsense reasoning.

Settings	OBQA	SIQA	WinoG	HellaS
GELU	88.8	83.5	88.5	95.8
ReLU	87.6	81.9	88.4	95.6
Tanh	87.4	82.4	87.5	94.5

Table 7: Comparison of different activation functions on four commonsense reasoning datasets.

4.6.3 Analysis of Rank Sensitivity

To assess RanLoRA's robustness and parameter efficiency, we analyze the rank r across eight commonsense reasoning benchmarks (Figure 2). Even at $r = 2$, RanLoRA achieves an average accuracy of 87.2%, outperforming most linear baselines with higher ranks, showing stability under extreme compression. Increasing r to 16 yields a modest gain to 87.9%, benefiting tasks like OBQA and BoolQ, while others such as ARC-e and HellaSwag plateau early. This demonstrates that our nonlinear residual mapping effectively captures task-specific manifolds without requiring high-dimensional linear subspaces. Across datasets from HellaSwag to BoolQ, RanLoRA maintains consistent performance, highlighting a favorable trade-off between parameter cost and expressiveness. Overall, our nonlinear progressive adaptation strategy achieves near-optimal performance with minimal parameters.

5 Conclusion

In this paper, we introduce RanLoRA, a proposed PEFT method that enhances representational capacity through nonlinear projection updates and by

using residual components of pretrained weights for initialization. Adopting a progressive fitting strategy, RanLoRA overcomes the linear subspace limitations of LoRA and its variants. Experiments across natural language understanding, commonsense and mathematical reasoning, and image classification show that RanLoRA achieves superior performance. In addition, it outperforms full fine-tuning and larger linear adapters with fewer parameters in several tasks, maintaining high expressiveness even under extreme rank compression. These results highlight the potential of nonlinear adaptation for large-scale foundation model optimization. We hope that RanLoRA will inspire further exploration of nonlinear extensions to LoRA.

Limitations

Despite its strong empirical performance and parameter efficiency, RanLoRA has several limitations that warrant further investigation. The use of Singular Value Decomposition (SVD) for decomposition introduces additional computational overhead. Although this cost is incurred only once prior to training, performing SVD on extremely large weight matrices may increase preprocessing time in ultra-large-scale models and affect practical scalability. While the proposed nonlinear projection substantially enhances expressiveness under low-rank constraints, the choice of the nonlinear activation function remains largely empirical. Different task domains and data distributions may favor different nonlinearities, and identifying principled guidelines for activation selection remains an open challenge. In addition, the empirical evaluation in this work primarily focuses on encoder-only (e.g., RoBERTa) and decoder-only (e.g., LLaMA) architectures. The effectiveness of RanLoRA for encoder–decoder models, as well as for other modalities such as speech or multimodal settings, has not been fully explored and is left for future work.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (No. 2025YFC2511700), the National Natural Science Foundation of China (Nos. 62576159 and 61402220), and the Natural Science Foundation of Hunan Province, China (Nos. 2025JJ50384 and 2022JJ30495).

References

Meta AI. 2024. The llama 3 model card.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, pages 7432–7439.

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

Gong Cheng, Junwei Han, and Xiaoqiang Lu. 2017. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. 2014. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, and 1 others. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature machine intelligence*, 5(3):220–235.

Haonan Dong, Wenhao Zhu, Guojie Song, and Liang Wang. 2025. Aurora: Breaking low-rank bottleneck of lora with nonlinear mapping. *arXiv preprint arXiv:2505.18738*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*.

Ziqi Gao, Qichao Wang, Aochuan Chen, Zijing Liu, Bingzhe Wu, Liang Chen, and Jia Li. 2024a. Parameter-efficient fine-tuning with discrete fourier transform. In *Forty-first International Conference on Machine Learning, ICML 2024,Vienna, Austria, July 21-27, 2024*.

Ziqi Gao, Qichao Wang, Aochuan Chen, Zijing Liu, Bingzhe Wu, Liang Chen, and Jia Li. 2024b. Parameter-efficient fine-tuning with discrete fourier transform. *Preprint*, arXiv:2405.03003.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and S Qian Zhang. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. arxiv 2024. *arXiv preprint arXiv:2403.14608*, 10.

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

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. 2019. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*.

Ting Jiang, Shaohan Huang, Shengyue Luo, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, Qi Zhang, Deqing Wang, and 1 others. 2024. Mora: High-rank updating for parameter-efficient fine-tuning. *arXiv preprint arXiv:2405.12130*.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in neural information processing systems*, 34:1022–1035.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 2013. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561.

Alex Krizhevsky. 2009. Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6:87–100.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Preprint*, arXiv:1907.11692.

Cui Long, Yongbin Liu, Chunping Ouyang, and Ying Yu. 2024. Bailicai: A domain-optimized retrieval-augmented generation framework for medical applications. *arXiv preprint arXiv:2407.21055*.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. 2013. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. Pissa: Principal singular values and singular vectors adaptation of large language models. *Advances in Neural Information Processing Systems*, 37:121038–121072.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

Rui Pan, Xiang Liu, Shizhe Diao, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. 2024. Lisa: Layerwise importance sampling for memory-efficient large language model fine-tuning. *Advances in Neural Information Processing Systems*, 37:57018–57049.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. 2012. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. Adapterfusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th conference of the European chapter of the association for computational linguistics: main volume*, pages 487–503.

Ruiyang Qin, Dancheng Liu, Chenhui Xu, Zheyu Yan, Zhaoxuan Tan, Zhenge Jia, Amir Nassereldine, Jiajie Li, Meng Jiang, Ahmed Abbasi, Jinjun Xiong, and Yiyu Shi. 2024. Empirical guidelines for deploying llms onto resource-constrained edge devices. *Preprint*, arXiv:2406.03777.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021. Adapterdrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 7930–7946.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8732–8740.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Lin Song, Yukang Chen, Shuai Yang, Xiaohan Ding, Yixiao Ge, Ying-Cong Chen, and Ying Shan. 2024. Low-rank approximation for sparse attention in multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13763–13773.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2023. Dylora: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3274–3287.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

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

Muling Wu, Wenhao Liu, Xiaohua Wang, Tianlong Li, Changze Lv, Zixuan Ling, Jianhao Zhu, Cenyuan Zhang, Xiaoqing Zheng, and Xuanjing Huang. 2024. Advancing parameter efficiency in fine-tuning via representation editing. *arXiv preprint arXiv:2402.15179*.

Taiqiang Wu, Jiahao Wang, Zhe Zhao, and Ngai Wong. 2025. Mixture-of-subspaces in low-rank adaptation. *Preprint*, arXiv:2406.11909.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

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

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. 2024. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*.

Yibo Zhong, Haoxiang Jiang, Lincan Li, Ryumei Nakada, Tianci Liu, Linjun Zhang, Huaxiu Yao, and Haoyu Wang. 2025. Neat: Nonlinear parameter-efficient adaptation of pre-trained models. *Preprint*, arXiv:2410.01870.

Hongyun Zhou, Xiangyu Lu, Wang Xu, Conghui Zhu, Tiejun Zhao, and Muyun Yang. 2025. Lora-drop: Efficient lora parameter pruning based on output evaluation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5530–5543.

A Baselines

To provide a comprehensive comparison of several PEFT methods, we briefly summarize several representative parameter-efficient fine-tuning (PEFT) approaches that are closely related to our work. These methods vary in their design choices, ranging from Low-Rank Adaptations and adapter modules to lightweight bias tuning and frequency-domain parameterization. Below we outline the core ideas of each method:

LORA. LoRA (Hu et al., 2022) inserts trainable low-rank matrices into the frozen pre-trained weights. This technique achieves efficient fine-tuning with fewer trainable parameters and less GPU memory usage.

AdaLoRA. AdaLoRA (Zhang et al., 2023) is an adaptive low-rank fine-tuning method that dynamically allocates parameter budget to assign optimal ranks to different weight matrices. Compared to LoRA with fixed rank, AdaLoRA achieves higher performance with the same number of parameters, improving fine-tuning efficiency and effectiveness.

DyLoRA. DyLoRA (Valipour et al., 2023)enables dynamic rank selection during inference by training a single LoRA module that supports all ranks up to a maximum. It shares parameters across ranks, allowing flexible trade-offs between efficiency and performance without retraining.

DoRA. DoRA (Liu et al., 2024) is an efficient fine-tuning method that decomposes pre-trained weights into magnitude and direction components, updating only the direction via Low-Rank Adaptation. It outperforms LoRA at the same parameter budget, especially in instruction-following and reasoning tasks.

Pissa. PiSSA (Meng et al., 2024) improves upon LoRA by initializing the adapter with principal singular values and vectors, optimizing the key components while freezing the "noisy" ones. This method leads to faster convergence and better performance than LoRA.

MiLoRA. MiLoRA (Wang et al., 2024) improves upon LoRA by initializing the adapter with minor singular values and vectors, optimizing the minor components while freezing the principal components. This method leads to faster convergence and better performance than LoRA. Meanwhile,it also reduce the knowledge forgetting.

MoSLoRA. MoSLoRA (Wu et al., 2025) introducing a learnable mixer to combine multiple low-rank subspaces, thereby improving adaptability and representational capacity.

NEAT. NEAT (Zhong et al., 2025) incorporates a lightweight neural network into the adaptation process.Unlike LoRA, which approximates weight updates linearly through low-rank decomposition, NEAT models cumulative weight updates as explicit functions of the pre-trained model's original weights.

LoRA-drop. LoRA-drop (Zhou et al., 2025) evaluates layer-wise LoRA importance via output changes on sampled task data, retains high-importance LoRA layers, shares LoRA across low-importance layers, and fine-tunes with fewer trainable parameters while minimizing performance loss.

AuroRA. AuroRA (Dong et al., 2025) is a non-linear parameter-efficient fine-tuning framework that compresses the latent rank r using MLP-style layers. It reduces parameter costs significantly while capturing high-order feature dynamics via learnable non-linearities, maintaining computational efficiency comparable to vanilla LoRA.

Adapter$^{\mathbf{H}}$. Adapter$^{\mathrm{H}}$ (Houlsby et al., 2019)inserts small bottleneck networks into each layer of a pretrained model and trains only these additional modules, enabling parameter-efficient adaptation to downstream tasks.

Adapter$^{\mathbf{P}}$. Adapter$^{\mathrm{P}}$ (Pfeiffer et al., 2021) is similar to Adapter$^{\mathrm{H}}$, the difference is that Adapter$^{\mathrm{P}}$ determines the adapter architecture that is optimal across all tasks through exhaustive search.

Adapter$^{\mathbf{D}}$. Adapter$^{\mathrm{D}}$ (Rücklé et al., 2021) is also similar to Adapter$^{\mathrm{H}}$,the modified part is that Adapter$^{\mathrm{D}}$ refers to applying dropout to the adapter modules and removing the adapters from the lower layers of the Transformer during both training and inference.

BitFit. BitFit (Zaken et al., 2021) updates only the bias terms of a pretrained model while freezing all other parameters, providing an extremely lightweight yet surprisingly effective parameter-efficient fine-tuning method.

FourierFT. FourierFT (Gao et al., 2024b) learns a small number of frequency-domain coefficients of weight updates via discrete Fourier transform, enabling parameter-efficient adaptation while maintaining model performance.

LP. LP(Linear probing) is a simple fine-tuning strategy where the pretrained model is frozen and only a lightweight linear classifier is trained on top of its representations.

B Datasets

B.1 Commonsense Reasoning

For commonsense reasoning task, we conduct evaluation on eight widely used benchmark datasets: BoolQ, PIQA, SIQA, HellaSwag, WinoGrande, ARC-e, ARC-c, and OBQA. (1) The BoolQ (Clark et al., 2019) dataset is a question-answering benchmark consisting of 15,942 examples, where the questions are naturally occurring and generated in unprompted and unconstrained settings, requiring yes/no answers. (2) The PIQA (Bisk et al., 2020) dataset presents questions with two potential solutions, demanding physical commonsense reasoning to identify the correct answer. (3) The SIQA (Sap et al., 2019) dataset focuses on reasoning about human actions and their social implications. (4) The HellaSwag (Zellers et al., 2019) dataset is designed for commonsense natural language inference (NLI) tasks, where each question includes a context and several potential endings, from which the correct continuation must be selected. (5) The WinoGrande (Sakaguchi et al., 2020) dataset is a fill-in-the-blank task with binary options, where the goal is to select the most plausible option for a given sentence requiring commonsense reasoning. (6) The ARC-c and (7) ARC-e (Clark et al., 2018) datasets refer to the Challenge and Easy sets, respectively, of the ARC dataset, which consists of multiple-choice science questions designed at a grade-school level, with the former being more challenging than the latter. (8) The OBQA (Mihaylov et al., 2018) dataset focuses on questions that necessitate multi-step reasoning, integration of external common knowledge, and in-depth text comprehension.The detailed of each dataset are summarized in Table 8.

Dataset	#Class	#Train	#Dev	#Test
BoolQ	Yes/No	9,427	3,270	3,245
PIQA	Option	16,113	1,838	3,000
SIQA	Option	33,410	1,954	2,224
HellaSwag	Option	39,905	10,042	10,003
WinoGrande	Option	40,398	1,267	1,767
ARC-e	Option	2,251	570	2,376
ARC-c	Option	1,119	229	1,172
OBQA	Option	4,957	500	500

Table 8: Detailed information of commonsense reasoning datasets.

B.2 Natural Language Understanding

The GLUE (Wang et al., 2018) benchmark comprises 8 NLP datasets: MNLI, SST-2, MRPC, CoLA, QNLI, QQP, RTE, and STS-B, covering tasks such as inference, sentiment analysis, paraphrase detection, linguistic acceptability, question-answering, and textual similarity. STS-B is a regression task, while all other tasks are either single-sentence or sentence-pair classification tasks. We provide detailed information about them in Table 9.

B.3 Image Classification

For image classification, we provide detailed information about the used datasets in Table 10.

B.4 Mathematical Reasoning

Detailed information for mathematical reasoning task is provided in Table 11. GSM8K consists of high quality grade school math problems, typically free-form answers.

C Experiment Settings

To ensure the reproducibility of our experimental results, we provide the detailed hyperparameter settings used in our experiments.Natural Language Understanding and image classification tasks run on four Tesla V100-PCIE-32GB GPUs. Commonsense reasoning and mathematical reasoning tasks run on a single H800 GPU (80GB).

C.1 Commonsense Reasoning

We provide hyperparameter settings of RanLoRA and other PEFT methods for commonsense reasoning task in Table 12. AuroRA (Dong et al., 2025), MoSLoRA (Wu et al., 2025), NEAT (Zhong et al., 2025), and LoRA (Hu et al., 2022) are following their own settings.

C.2 Natural Language Understanding

We provide used hyper-parameters for RanLoRA in natural language understanding on the GLUE benchmark in Table 13. NEAT (Zhong et al., 2025) and AuroRA's (Dong et al., 2025) are following their papers' settings.

C.3 Image Classification

Hyperparameter for RanLoRA are provided in Table 14. We tune the classification head and the backbone separately and provide detailed settings for each dataset. The scaling factor s is set to 2.0. The rank r for MHSA is set to 2 in the QV-setting. NEAT (Zhong et al., 2025) and AuroRA's (Dong et al., 2025) are following their papers' settings.

Corpus	Task	Metric	# Train	# Val	# Test	# Labels
Single-Sentence Tasks						
CoLA	Acceptability	Matthews Corr.	8.55k	1.04k	1.06k	2
SST-2	Sentiment	Accuracy	67.3k	872	1.82k	2
Similarity and Paraphrase Tasks						
MRPC	Paraphrase	Accuracy / F1	3.67k	408	1.73k	2
STS-B	Sentence similarity	Pearson / Spearman Corr.	5.75k	1.5k	1.38k	1
QQP	Paraphrase	Accuracy / F1	364k	40.4k	391k	2
Inference Tasks						
MNLI	NLI	Accuracy	393k	19.65k	19.65k	3
QNLI	QA / NLI	Accuracy	105k	5.46k	5.46k	2
RTE	NLI	Accuracy	2.49k	277	3k	2

Table 9: Detailed information of the GLUE benchmark.

Dataset	#Class	#Train	#Val	#Test	Rescaled resolution
OxfordPets	37	3,312	368	3,669	224×224
StandfordCars	196	7,329	815	8,041	
CIFAR10	10	45,000	5,000	10,000	
DTD	47	4,060	452	1,128	
EuroSAT	10	16,200	5,400	5,400	
FGVC	100	3,000	334	3,333	
RESISC45	45	18,900	6,300	6,300	
CIFAR100	100	45,000	5,000	10,000	

Table 10: Detailed information of image classification datasets.

Dataset	#Train	#Dev	#Test
GSM8K	7,473	1,319	1,319

Table 11: Detailed information of mathematical reasoning task.

C.4 Mathematical Reasoning

We provide hyperparameter settings of RanLoRA and AuroRA (Dong et al., 2025) for mathematical reasoning task in Table 15. We limit all samples to a maximum of 768 tokens. For evaluation, we set a maximum token number of 256 on GSM8K (Cobbe et al., 2021) dataset.

D Additional Experiments

To further validate the scalability of RanLoRA, we conduct extensive experiments on the LLaMA3-8B model across eight commonsense reasoning tasks. We compare RanLoRA under different initialization schemes and activation functions against the AuroRA baseline, as shown in Table 16. We could observe that:

Superiority over Nonlinear Baselines. The results show that RanLoRA consistently outperforms AuroRA across all rank settings. While AuroRA (K&T) reaches a peak average accuracy of 84.5% at $r = 16$, RanLoRA achieves 87.2% even at the minimum rank of $r = 2$ with R&G. This 2.7% gain at 1/8 of the parameter cost highlights the efficiency of our nonlinear residual mapping.

Synergy of Residual-aware Initialization and Nonlinearity. Proper initialization is crucial for effective adaptation. Methods leveraging SVD-based initialization—either using the top-r principal components or the residual components of pretrained weights—consistently outperform standard Kaiming initialization as in AuroRA. Within RanLoRA, combining residual-aware initialization with a GELU nonlinearity achieves the best performance, reaching an average accuracy of 87.9% at $r = 16$, highlighting the synergistic effect of residual alignment and nonlinear transformation.

Robustness of Activation Functions. Across all SVD initialization methods, **GELU** consistently outperforms ReLU and Tanh. For example, under PiSSA at $r = 2$, GELU exceeds ReLU by 0.3% and Tanh by 0.5%. This confirms that GELU's smooth, innate gating mechanism better captures complex in large-scale decoders.

Hyperparameter	RanLoRA(GELU)	RanLoRA(ReLU)	RanLoRA(Tanh)	AuroRA	LoRA	MoSLoRA	NEAT
Optimizer	AdamW						
Dropout	0.05						
Batch size	16						
Target module	q,k,v,up,down						
Warmup steps	100						
Epochs	3						
Rank $\hat{r}$	2,4,8,16						
α	$2\hat{r}$						
Learning rate	1e-4	3e-4	3e-5	3e-4	3e-4	3e-4	3e-4

Table 12: Hyperparameter of commonsense reasoning for RanLoRA and other PEFT methods.

Model	Hyperparameter	STS-B	RTE	MRPC	CoLA	SST-2	QNLI
Both	Optimizer	AdamW					
	LR Schedule	Linear					
	Warmup Ratio	0.06					
	Rank $\hat{r}$	2					
	α	4					
Base	Learning Rate	4e-4	3e-4	2e-4	8e-4	6e-4	4e-4
	Max Seq. Len	512	512	512	512	512	512
	Batch Size	64	32	64	64	32	32
	Epochs	30	30	40	80	50	40
Large	Learning Rate	4e-4	2e-3	4e-4	7e-4	8e-4	2e-4
	Max Seq. Len	512	512	512	512	512	512
	Batch Size	64	64	64	64	64	32
	Epochs	20	30	40	40	50	20

Table 13: Hyperparameter settings across GLUE benchmark for RanLoRA.

E The Use of LLMs

This manuscript has been polished with the assistance of LLMs, which were used solely for language refinement and not for ideation or substantive writing.

Model	Hyperparameter	OxfordPets	StanfordCars	CIFAR10	DTD	EuroSAT	FGVC	RESISC45	CIFAR100
Both	Epochs	10							
	Optimizer	AdamW							
	LR Schedule	Linear							
	Rank $\hat{r}$	2							
	α	4							
	Weight Decay	8e-4	4e-5	9e-5	7e-5	3e-4	7e-5	3e-4	1e-4
Base	Learning Rate (Head)	5e-3	1e-2	5e-3	1e-2	5e-3	1e-2	1e-2	5e-3
	Learning Rate (RanLoRA)	4e-3	1e-2	4e-3	7e-3	8e-3	8e-3	7e-3	3e-3
Large	Learning Rate (Head)	5e-3	1e-2	5e-3	4e-3	5e-3	1e-2	1e-2	5e-3
	Learning Rate (RanLoRA)	5e-3	9e-3	2e-3	9e-3	5e-3	8e-3	6e-3	4e-3

Table 14: Hyperparameters for image classification for RanLoRA.

Hyperparameter	RanLoRA(GELU)	AuroRA
Optimizer	AdamW	
Dropout	0.05	
Batch size	16	
Target module	q,k,v,up,down	
Warmup steps	100	
Epochs	3	
Rank $\hat{r}$	2,4,8	
α	$2\hat{r}$	
Learning rate	1e-4	3e-4

Table 15: Hyperparameters of mathematical reasoning for RanLoRA and AuroRA.

Init&Act	Method	Param	ARC-e	OBQA	SIQA	ARC-c	WinoG	PIQA	BoolQ	HellaS	Avg
K&T	AuroRA$_{r=2}$	3.5M	89.4	84.8	79.7	78.3	83.9	87.1	73.5	94.3	83.9
	AuroRA$_{r=4}$	7.0M	89.1	84.0	79.7	79.4	84.8	87.9	72.7	94.7	84.0
	AuroRA$_{r=8}$	14.1M	89.3	85.4	80.3	78.6	85.5	88.2	74.0	95.0	84.5
	AuroRA$_{r=16}$	28.3M	89.1	84.0	81.0	79.2	85.2	88.2	74.1	94.8	84.5
P&R	RanLoRA$_{r=2}$	3.5M	92.8	87.2	82.4	83.5	88.3	89.1	73.5	95.1	86.5
	RanLoRA$_{r=4}$	7.0M	93.6	88.2	81.9	84.0	88.2	89.3	73.9	95.3	86.8
	RanLoRA$_{r=8}$	14.1M	92.5	88.6	83.1	82.3	88.2	89.1	75.0	94.9	86.7
	RanLoRA$_{r=16}$	28.3M	93.0	89.6	82.9	81.8	88.2	89.4	74.5	94.4	86.7
P&T	RanLoRA$_{r=2}$	3.5M	92.8	87.8	81.9	82.5	87.8	89.1	74.7	93.8	86.3
	RanLoRA$_{r=4}$	7.0M	92.9	89.0	81.8	82.0	88.1	89.7	74.1	94.5	86.5
	RanLoRA$_{r=8}$	14.1M	92.4	87.4	82.2	82.0	88.9	89.7	74.2	94.3	86.4
	RanLoRA$_{r=16}$	28.3M	92.8	85.6	81.2	81.7	88.2	88.8	74.3	94.8	86.0
P&G	RanLoRA$_{r=2}$	3.5M	93.1	88.8	81.9	82.5	87.8	89.2	75.4	95.4	86.8
	RanLoRA$_{r=4}$	7.0M	93.9	88.6	81.4	83.2	89.3	89.4	74.3	95.5	87
	RanLoRA$_{r=8}$	14.1M	93.9	87.8	82.7	83.7	88.6	90.8	74.5	95.8	87.2
	RanLoRA$_{r=16}$	28.3M	93.1	90.6	81.6	84.1	89	89.6	74.8	95.2	87.3
R&R	RanLoRA$_{r=2}$	3.5M	92.9	87.6	81.9	83.2	88.4	88.8	75.5	95.6	86.7
	RanLoRA$_{r=4}$	7.0M	92.4	88.6	81.3	82.0	88.0	89.2	75.8	95.3	86.6
	RanLoRA$_{r=8}$	14.1M	93.3	87.4	81.1	82.0	88.0	88.9	72.5	94.7	86.0
	RanLoRA$_{r=16}$	28.3M	93.1	87.8	81.8	82.2	88.2	88.4	73.4	94.6	86.2
R&T	RanLoRA$_{r=2}$	3.5M	92.6	87.4	82.4	81.2	87.5	90.3	73.7	94.5	86.2
	RanLoRA$_{r=4}$	7.0M	93.3	89.2	82.1	83.0	88.8	90.1	74.7	94.8	87.0
	RanLoRA$_{r=8}$	14.1M	93.6	88.2	82.4	82.5	88.8	90.8	74.1	95.2	87.0
	RanLoRA$_{r=16}$	28.3M	93.3	87.8	82.5	83.3	88.7	90.2	76.6	95.6	87.3
R&G	RanLoRA$_{r=2}$	3.5M	93.1	88.8	83.5	83.1	88.5	89.8	75.1	95.8	87.2
	RanLoRA$_{r=4}$	7.0M	93.6	88.4	82.2	83.6	88.0	90.9	75.0	95.3	87.1
	RanLoRA$_{r=8}$	14.1M	93.1	90.6	81.6	84.0	90.3	90.6	75.3	95.6	87.6
	RanLoRA$_{r=16}$	28.3M	93.1	91.4	82.7	84.6	90.0	89.9	75.9	95.9	87.9

Table 16: Commonsense reasoning performance of RanLoRA and AuroRA on LLaMA3-8B with different initialization and activation settings. "Init&Act" denotes the combination of initialization method and activation function, where (a)&(b) indicates initialization with method "a" and activation function "b". Initialization methods are: K (Kaiming), P (Principal-Aware initialization,which uses top-r principal components of pretrained weights), and R (Residual-aware initialization, which uses residual r components of pretrained weights). Activation functions are: T (Tanh), R (ReLU), and G (GELU). "Avg." refers to the average accuracy across all datasets.