Polynomial Expansion Rank Adaptation: Enhancing Low-Rank Fine-Tuning with High-Order Interactions

Wenhao Zhang[1] **and Lin Mu**[1*] **and Li Ni**[1] **and Peiquan Jin**[2] **and Yiwen Zhang**[1]
[1] Anhui University,
[2] University of Science and Technology of China,
{mulin, nili, zhangyiwen}@ahu.edu.cn zhangwenhao@stu.ahu.edu.cn
jpq@ustc.edu.cn

Abstract

Low-rank adaptation (LoRA) is a widely used strategy for efficient fine-tuning of large language models (LLMs), but its strictly linear structure fundamentally limits expressive capacity. The bilinear formulation of weight updates captures only first-order dependencies between low-rank factors, restricting the modeling of nonlinear and higher-order parameter interactions. In this paper, we propose **P**olynomial **E**xpansion **R**ank **A**daptation (**PERA**), a novel method that introduces structured polynomial expansion directly into the low-rank factor space. By expanding each low-rank factor to synthesize high-order interaction terms before composition, PERA transforms the adaptation space into a polynomial manifold capable of modeling richer nonlinear coupling without increasing rank or inference cost. We provide theoretical analysis demonstrating that PERA offers enhanced expressive capacity and more effective feature utilization compare to existing linear adaptation approaches. Empirically, PERA consistently outperforms state-of-the-art methods across diverse benchmarks. Notably, our experiments show that incorporating high-order nonlinear components—particularly square terms—is crucial for enhancing expressive capacity and maintaining strong and robust performance under various rank settings. Our code is available at https://github.com/zhangwenhao6/PERA

1 Introduction

Large language models (Brown et al., 2020; Ouyang et al., 2022; Touvron et al., 2023b) (LLMs) are typically trained on large-scale general-domain datasets using the next-token prediction objective (Brown et al., 2020), and exhibit remarkable generalization capabilities across a wide range of natural language processing tasks (Thirunavukarasu et al., 2023; Ruiz et al., 2023). However, as model sizes continue to grow and downstream tasks become increasingly diverse, the computational and memory costs associated with full fine-tuning have risen dramatically (Touvron et al., 2023a; Jiang et al., 2023), thereby constraining the widespread deployment of LLMs.

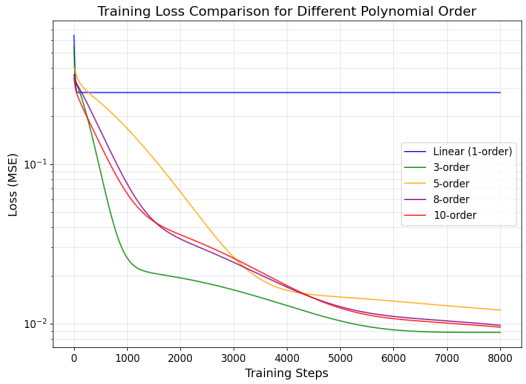


Figure 1: Comparison of Training Losses between First-Order and High-Order Terms.

To address this challenge, parameter-efficient fine-tuning (PEFT) (Houlsby et al., 2019; Liu et al., 2021; Lialin et al., 2023) has emerged as an effective paradigm that adapts LLMs by introducing a small number of task-specific parameters while keeping the backbone weights frozen. Among various PEFT methods, low-rank adaptation (LoRA) (Hu et al., 2022) achieves strong efficiency by restricting weight updates to a low-rank subspace. Nevertheless, LoRA's strictly bilinear update form $\Delta W = BA$ inherently restricts the expressivity of the adaptation, as it captures only first-order linear dependencies between the low-rank factors, restricting its ability to model multi-dimensional or high-order dependencies (Liu et al., 2023; Jiang et al., 2024; Zhuang et al., 2024). To partially mitigate this, HiRA (Huang et al., 2025) introduces Hadamard modulation with pretrained weights, enriching multi-dimensional representa-

*Corresponding author

Findings of the Association for Computational Linguistics: ACL 2026, pages 13287–13303

tion. However, its update mechanism remains fundamentally linear with respect to trainable parameters and depends on external weight coupling, thus offering limited ability to model intrinsic high-order relations.

From the perspective of function approximation, there exists a fundamental difference in expressive capacity within function space between a first-order linear function $f(x) = c + c_1 x$ and a polynomial function with higher-order terms $f(x) = c + c_1 x + c_2 x^2 + \cdots + c_n x^n$, as illustrated in Fig. 1. This directly leads to significant differences in fitting accuracy, convergence speed, and training loss between the two. If the traditional LoRA mapping is viewed as a first-order linear approximation of weight updates, its limitation in expressive capacity becomes immediately evident.

Motivated by these observation, we investigate whether the expressive capacity of low-rank adaptation can be enhanced by introducing explicit high-order feature interactions and structured nonlinearity directly within the low-rank factor space—without increasing rank or depending on external modulation. Drawing inspiration from classical polynomial feature expansion (Kuhn and Johnson, 2019; Galli, 2020; Ozdemir, 2022), which systematically constructs higher-order feature terms from base representations, we propose **P**olynomial **E**xpansion **R**ank **A**daptation (**PERA**).

Unlike traditional methods that perform polynomial expansion in the feature space, PERA applies a structured polynomial mapping directly within the parameter space of low-rank factors. Specifically, before composition, each low-rank matrix is expanded along its column dimension to generate higher-order parameter interaction terms. This operation constructs a polynomial manifold in the adaptation space, enabling ΔW to capture richer nonlinear coupling relationships among adaptation directions. Empirical analyses demonstrate faster convergence and reduced training loss. Furthermore, our theoretical results indicate that this formulation significantly enhances both the expressive capacity of adaptation and the efficiency of feature utilization. Notably, we implement higher-order interaction terms via matrix concatenation rather than sequential matrix addition. As a result, no additional inference overhead is introduced, while preserving the modular efficiency of LoRA.

Our contributions can be summarized as follows:

- We propose PERA that introduces polynomial expansion within the parameter space of low-rank factors. PERA explicitly models high-order interactions and structured nonlinearity among adaptation directions, enhancing representational expressivity without increasing rank or parameter cost.
- We theoretically show that parameter-space polynomial expansion improves both the expressive capacity and feature utilization efficiency of low-rank adaptation, offering a principled explanation for its stronger representation power.
- We empirically demonstrate that, compared to other PEFT methods, our approach maintains a computational and memory footprint close to stan-dard LoRA and achieves lower training loss.

2 Related Work

Parameter Efficient Fine-Tuning. Parameter-efficient fine-tuning (PEFT) aims to substantially reduce the number of trainable parameters while maintaining model performance and efficiency. To this end, researchers have explored a variety of approaches. For instance, Adapter (Hu et al., 2023) inserted lightweight feed-forward network modules into Transformer (Vaswani et al., 2017) layers and updates only these newly introduced parameters for downstream adaptation. Prefix-tuning (Li and Liang, 2021) learned continuous prefix vectors that encode task-specific information and steer the model toward generating task-relevant outputs. Prompt-tuning (Lester et al., 2021) further simplified this paradigm by optimizing only the embedding representations of input prompts, thereby effectively activating the model's capabilities on downstream tasks. Despite their strong performance, these methods typically introduce additional computational overhead during inference.

Low-Rank Adaptation (LoRA). LoRA (Hu et al., 2022) leverages the observation that weight updates in pretrained models have low intrinsic rank. It introduces two trainable low-rank matrices, A and B, into linear layers to approximate task-specific updates without altering the architecture or adding inference latency. Given a pretrained weight W_0, the adapted weight is defined as:

$$W' = W_0 + \Delta W = W_0 + \frac{\alpha}{r} BA, \qquad (1)$$

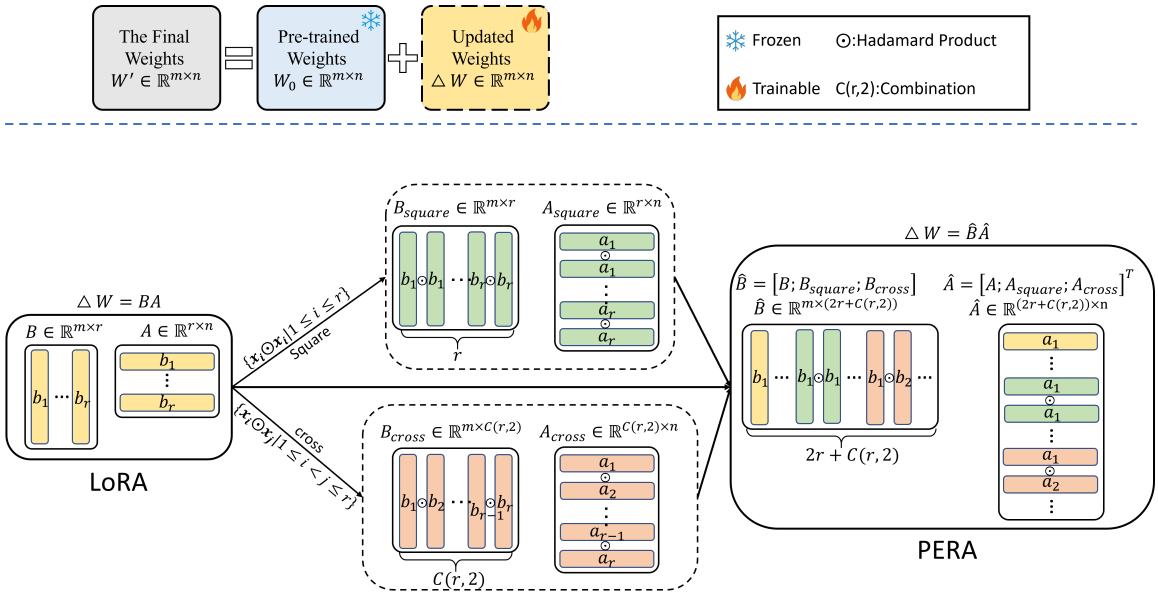


Figure 2: The architecture comparison between LoRA and PERA. By applying the polynomial expansion technique, PERA synthesizes new high-order feature terms from the original r feature terms in the low-rank matrices, including r **square feature terms** and $C(r,2)$ **crossed feature terms.**

where α is a scaling factor and r denotes the rank, with $r \ll \min\{m,n\}$. During fine-tuning, the pre-trained weights W_0 remain frozen, and the model is adapted by optimizing the low-rank matrices A and B. Building upon LoRA, DoRA(Liu et al., 2024) decomposed the weight updates into magnitude and directional components, applying normalization and scaling to each singular column of W'. Similarly, DeLoRA (Bini et al., 2025) performed normalization within the internal r-dimensional space of each low-rank matrix. To learn high-rank weight updates, MoRA(Jiang et al., 2024) compressed, transformed, and decompressed the input to enable high-rank adaptation, while HiRA(Huang et al., 2025) increased the maximum achievable rank by applying the Hadamard product between the updated weights ΔW and the original weights W.

Unlike HiRA and LoRA, PERA employs polynomial expansion to generate explicit high-order feature terms from low-rank factors. This design enhances nonlinear expressivity and cross-dimensional interactions while maintaining LoRA's efficiency and zero inference overhead.

3 Methodology

3.1 Preliminaries: Polynomial Expansion

Polynomial expansion (Kuhn and Johnson, 2019; Galli, 2020; Ozdemir, 2022) is a commonly used technique in feature engineering, primarily employed to generate synthetic features from original representations when feature dimensionality is limited.

Formally, consider a matrix $B \in \mathbb{R}^{m\times r}$ whose column vectors are denoted as $B = [\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_r]$, where each $\mathbf{b}_i \in \mathbb{R}^m$. We define a k-th order polynomial expansion operation as $\text{Poly}^k(B)$, which constructs all possible k-order polynomial combinations among the columns of B. For computational efficiency, we typically adopt the 2-order polynomial expansion, formula as:

$$\text{Poly}^2(B) = [\mathbf{b}_i, \mathbf{b}_i \odot \mathbf{b}_j \mid 1 \le i \le j \le r] \quad (2)$$

where $\odot$ denotes element-wise multiplication (Hadamard product). The expanded feature set can be written as $\hat{B} = [B; B_{square}; B_{cross}]$, consisting of r original features, r square features and $C(r,2)$ pairwise cross features:

$$\begin{aligned} B &= \{\mathbf{b}_i \mid 1 \le i \le r\}, \\ B_{square} &= \{\mathbf{b}_i \odot \mathbf{b}_j \mid 1 \le i = j \le r\}, \\ B_{cross} &= \{\mathbf{b}_i \odot \mathbf{b}_j \mid 1 \le i < j \le r\} \end{aligned}$$

This expansion maps the original low-rank space $\mathbb{R}^r$ to a higher dimensional space $\mathbb{R}^{2r+C(r,2)}$.

To further enhance representation capacity, we extend this formulation by introducing a Hadamard-based polynomial expansion. Similarly, consider a matrix $A \in \mathbb{R}^{r\times n}$ whose row vectors are represented as $A = [\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_r]^T$, where each

$\mathbf{a}_i \in \mathbb{R}^n$. We define the Hadamard-based polynomial expansion as:

$$\text{Poly}_{\text{H}}^2(A) = [\mathbf{a}_i, h_{ij}(\mathbf{a}_i \odot \mathbf{a}_j) \mid 1 \le i \le j \le r]^T. \quad (3)$$

Accordingly, this expansion can be defined as $\hat{A} = [A; A_{square}; A_{cross}]^T$:

$$\begin{aligned} A &= \{\mathbf{a}_i \mid 1 \le i \le r\}, \\ A_{square} &= \{h_{ij}(\mathbf{a}_i \odot \mathbf{a}_j) \mid 1 \le i = j \le r\}, \\ A_{cross} &= \{h_{ij}(\mathbf{a}_i \odot \mathbf{a}_j) \mid 1 \le i < j \le r\} \end{aligned}$$

here, $\mathbf{h} = \{h_{ij} \mid 1 \le i \le j \le r\}$ is learnable coefficients initialized to zero for stability. This initialization strategy ensures smooth optimization while allowing the model to gradually incorporate nonlinear contributions from high-order features.

3.2 PERA Architecture

We propose PERA, a simple yet effective PEFT method that introduces polynomial expansion into the low-rank adaptation space. Inspired by polynomial techniques in feature engineering, PERA enhances the expressive power of weight updates by enabling explicit high-order interactions among low-rank factors—without increasing the nominal rank. Following the LoRA framework, we decompose the weight update into two trainable low-rank matrices $B \in \mathbb{R}^{m\times r}$ and $A \in \mathbb{R}^{r\times n}$, initialized with a Gaussian distribution and zero values, respectively. Then, we apply structured polynomial mappings to these matrices: For B, we perform a standard polynomial expansion $\text{Poly}^2(B)$ (Eq. 2). For A, we employ the Hadamard-based polynomial expansion $\text{Poly}_{\text{H}}^2(A)$ (Eq. 3) to ensure stability during optimization. The resulting parameter update is defined as:

$$\Delta W = \hat{B}\hat{A} = \text{Poly}^2(B)\text{Poly}_{\text{H}}^2(A), \quad (4)$$

where, $\text{Poly}^2(B) \in \mathbb{R}^{m\times(2r+C(r,2))}$ denotes the expanded matrix obtained by applying polynomial expansion to the low-rank matrix B, and $\text{Poly}_{\text{H}}^2(A) \in \mathbb{R}^{(2r+C(r,2))\times n}$ represents the matrix produced by applying the Hadamard-based polynomial expansion to the low-rank matrix A.

We integrate the frozen pretrained weight W_0 with the adapted hidden representation. Specifically, for a linear layer with forward computation $h = W_0 x$, where $x \in \mathbb{R}^n$ denotes the input representation, PERA modifies its forward computation as follows:

$$\begin{aligned} h &= W'x \\ &= W_0 x + \text{Poly}^2(B)\,\text{Poly}_{\text{H}}^2(A)\,x. \end{aligned} \quad (5)$$

Notably, during backpropagation, gradients are computed only with respect to the low-rank matrices A and B, along with the Hadamard vector $\mathbf{h}$. As a result, compared to the LoRA, PERA does not introduce significant additional parameter overhead while substantially improving expressive capacity.

3.3 Analysis of PERA

3.3.1 Rank Analysis

LoRA and its variants construct the weight update ΔW as the product of two low-rank matrices, which inherently limits the maximum achievable rank of the updated weight. Let the pretrained weight matrix be $W_0 \in \mathbb{R}^{m\times n}$ with rank r_0, where $r_0 \le min(m, n)$. According to Eq. 1 and basic properties of matrix rank, the rank of the updated weight $W' = W_0 + \Delta W$ satisfies:

$$\text{Rank}(W') \le r_0 + r. \quad (6)$$

This implies that the low-rank structure of ΔW restricts the rank growth of W', potentially limiting the expressiveness of the adapted model. In contrast, PERA applies polynomial expansion to the low-rank factors, resulting in an expanded factor dimension of $2r + C(r, 2)$(Eq. 5). Consequently, the rank of the adapted weight in PERA is upper-bounded by:

$$\text{Rank}(W') \le r_0 + (2r + C(r, 2)). \quad (7)$$

Compared with Eq. 6, PERA substantially increases the theoretical upper bound on the rank of the adapted weight matrix, thereby enlarging the space of feasible updates and enabling more expressive adaptation under the PEFT framework. A detailed proof is provided in Appendix A.1.

3.3.2 Feature Utilization Analysis

Another key difference between PERA and LoRA lies in the expressive form of the weight update. In LoRA, the update matrix is constructed as the product of two low-rank matrices $B \in \mathbb{R}^{m\times r}$ and $A \in \mathbb{R}^{r\times n}$. By representing B and A in terms of their rank-one components: $B = [\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_r]$, the LoRA update can be expressed as:

$$\Delta W = BA = \sum_{i=1}^{r} \mathbf{b}_i \mathbf{a}_i^T, \quad (8)$$

which is a linear combination of rank-one matrices.

In contrast, PERA enriches the low-rank update by introducing polynomial feature expansions on the low-rank factors. Specifically, PERA incorporates both square feature terms and pairwise cross terms through polynomial expansion of B and a Hadamard-based polynomial expansion of A. As a result, the update matrix in PERA can be expressed as:

$$\begin{aligned}\Delta W &= \text{Poly}^2(B)\text{Poly}^2_{\text{H}}(A) \\ &= \sum_{i=1}^{r} \mathbf{b}_i \mathbf{a}_i^T \\ &\quad + \sum_{1\le i=j}^{r} h_{ij}(\mathbf{b}_i \odot \mathbf{b}_j)(\mathbf{a}_i^T \odot \mathbf{a}_j^T) \\ &\quad + \sum_{1\le i<j}^{r} h_{ij}(\mathbf{b}_i \odot \mathbf{b}_j)(\mathbf{a}_i^T \odot \mathbf{a}_j^T).\end{aligned} \tag{9}$$

Comparing Eq. 8 and Eq. 9, we observe that PERA explicitly augments the linear low-rank update with square and cross feature terms, introducing structured high-order nonlinear components into ΔW. This enriched formulation enables more diverse feature utilization and substantially enhances the expressive capacity of the adaptation, while preserving the factorized structure of low-rank updates. A detailed theoretical analysis is provided in Appendix A.2.

3.3.3 Relation between LoRA and PERA.

We further observe that LoRA can be regarded as a special case of PERA. Concretely, PERA follows the same initialization strategy as LoRA, where the two low-rank matrices A and B are initialized with a Gaussian distribution and zero values. Based on Eq. 8 and Eq. 9, when the Hadamard vector $\mathbf{h}$ is initialized to zero values and kept frozen during training, denoted as $\{frozen(h_{ij} = 0) \mid 1 \le i \le j \le r\}$, PERA becomes equivalent to LoRA.

Furthermore, when only the cross coefficients are initialized to zero values and kept frozen, represented as $\{frozen(h_{ij}) = 0) \mid 1 \le i < j \le r\}$, PERA becomes a LoRA variant that introduces only high-order square feature terms:

$$\begin{aligned}\Delta W &= \text{Poly}^2(B)\text{Poly}^2_{\text{H}}(A) \\ &= \sum_{i=1}^{r} \mathbf{b}_i \mathbf{a}_i^T \\ &\quad + \sum_{1\le i=j}^{r} h_{ij}(\mathbf{b}_i \odot \mathbf{b}_j)(\mathbf{a}_i^T \odot \mathbf{a}_j^T).\end{aligned} \tag{10}$$

Similarly, when only the square coefficients are initialized to zero values and kept frozen, denoted as $\{frozen(h_{ij} = 0) \mid 1 \le i = j \le r\}$, PERA becomes a LoRA variant that incorporates high-order cross feature terms:

$$\begin{aligned}\Delta W &= \text{Poly}^2(B)\text{Poly}^2_{\text{H}}(A) \\ &= \sum_{i=1}^{r} \mathbf{b}_i \mathbf{a}_i^T \\ &\quad + \sum_{1\le i<j}^{r} h_{ij}(\mathbf{b}_i \odot \mathbf{b}_j)(\mathbf{a}_i^T \odot \mathbf{a}_j^T).\end{aligned} \tag{11}$$

A detailed analysis of the impact of LoRA and its high-order variant can be found in Section 5.4

4 Experiment

In this section, we conduct a systematic evaluation of the PERA across multiple tasks, including commonsense reasoning, natural language understanding(NLU). ALL experiments are conducted on NVIDIA RTX 5090 GPUs.

4.1 Commonsense Reasoning

Models and Datasets. We fine-tune LLaMA2-7B (Touvron et al., 2023a) and LLaMA3-8B (Grattafiori et al., 2024) models on the commonsense170K dataset (Hu et al., 2023). This dataset integrates eight commonsense reasoning benchmarks, 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., 2021), ARC-c and ARC-e (Clark et al., 2018), and OBQA (Mihaylov et al., 2018).

Implementation Details. To ensure a fair comparison, we strictly follow the experimental setup described in (Huang et al., 2025). Specifically, when fine-tuning the LLaMA series models, we adjust only the learning rate to $1e-4$, while keeping all other hyperparameters identical in (Huang et al., 2025). For final evaluation, we select the checkpoint that achieves the best performance on the validation set. Detailed training hyperparameters can be found in the Appendix B.2.

Main Results. As shown in Table 1, PERA consistently demonstrates stable and superior performance across all eight commonsense reasoning tasks on both LLaMA2-7B and LLaMA3-8B. On LLaMA2-7B, PERA achieves an average accuracy of 82.61%, outperforming LoRA (77.61%) by 5%.

Model	Method	#Params (%)	BoolQ	PIQA	SIQA	ARC-c	ARC-e	OBQA	HellaS	WinoG	Avg.
ChatGPT	-	-	73.10	85.40	68.50	79.90	89.80	74.80	78.50	66.10	77.01
Llama-2-7B	Prompt Tuning	0.0012	55.93	12.35	30.50	6.06	8.63	9.40	6.91	40.57	21.29
	P-Tuning	0.7428	58.75	36.02	0.20	0.17	1.98	0.80	0.01	0.00	12.24
	LoRA ($r = 32$)	0.8256	69.80	79.90	79.50	64.70	79.80	81.00	83.60	82.60	77.61
	DoRA ($r = 32$)	0.8256	71.80	83.70	76.00	68.20	83.70	82.40	**89.10**	82.60	79.69
	MoRA ($r = 32$)	0.8241	72.17	80.79	79.53	71.42	85.31	81.20	29.09	80.19	72.46
	HiRA ($r = 32$)	0.8256	71.22	83.35	79.53	73.81	86.74	84.60	88.12	83.98	81.42
	PERA ($r = 16$)	0.4148	71.83	**85.31**	80.35	**73.55**	**88.55**	**85.40**	88.66	**87.13**	**82.61**
	PERA ($r = 32$)	0.8268	**73.70**	84.39	79.53	73.29	87.79	83.20	88.98	86.27	82.14
Llama-3-8B	Prompt Tuning	0.0010	56.85	45.05	36.13	31.57	32.74	29.20	14.01	50.12	36.96
	P-Tuning	0.6240	59.97	11.64	8.19	7.42	8.63	9.60	1.77	37.65	18.11
	LoRA ($r = 16$)	0.3513	72.30	86.70	79.30	75.70	87.70	82.80	93.50	84.80	82.80
	LoRA ($r = 32$)	0.7002	70.80	85.20	79.90	71.20	84.20	79.00	91.70	84.30	80.79
	DoRA ($r = 32$)	0.7002	74.60	89.30	79.90	80.40	90.50	85.80	95.50	85.60	85.20
	MoRA ($r = 32$)	0.6997	74.28	87.43	80.71	79.61	91.16	85.60	43.53	86.74	78.63
	HiRA ($r = 16$)	0.3513	73.85	89.12	81.06	82.59	93.06	87.40	94.85	86.74	86.08
	HiRA ($r = 32$)	0.7002	75.40	89.70	81.15	82.90	93.27	88.32	95.36	87.70	86.72
	PERA ($r = 16$)	0.3515	**76.15**	88.57	**83.16**	**83.62**	**93.73**	**88.60**	95.94	**89.27**	**87.38**
	PERA ($r = 32$)	0.7012	75.54	89.83	81.58	83.53	93.56	88.40	95.61	88.95	87.12
Qwen2.5-7B	LoRA ($r = 16$)	0.3541	60.00	73.60	70.00	71.70	85.90	74.40	78.60	75.80	73.80
	HiRA ($r = 16$)	0.3541	69.00	88.30	80.80	88.70	95.40	88.00	92.30	81.00	85.40
	PERA ($r = 16$)	0.3543	**73.06**	**89.61**	**82.04**	**88.91**	**96.17**	**91.80**	**95.38**	**89.34**	**88.29**

Table 1: Accuracy(%) comparison of LLaMA model with various fine-tuned methods on commonsense reasoning datasets. Results of all baseline methods are taken from (Wu et al., 2024b; Huang et al., 2025). The best performance within each LLM is indicated in bold, while the second best performance is highlighted in underline.

On LLaMA3-8B, PERA reaches 87.38%, surpassing the best baseline method HiRA. These results suggest that PERA provides a more expressive and effective adaptation mechanism for complex reasoning tasks.

Notably, PERA maintains its performance advantages even under extremely low-rank settings. For instance, with rank $r = 4$, PERA achieves average accuracies of 81.57% and 87.01%, which are close to its best performance at $r = 16$. This robustness under strict parameter constraints indicates that PERA can effectively exploit low-rank parameters to construct richer adaptation representations, leading to superior performance compared to conventional low-rank adaptation methods.

4.2 Natural Language Understanding

Models and Datasets. We evaluate the effectiveness of PERA for adapting small language models by fine-tuning the RoBERTa-base model (Zhuang et al., 2021) on the GLUE (General Language Understanding Evaluation) benchmark (Wang et al., 2018). The GLUE benchmark is widely used to assess natural language understanding capabilities and comprises a diverse set of tasks, including natural language inference, sentiment classification and textual entailment. We conduct experiment on six datasets from the GLUE benchmark: SST-2, MRPC, CoLA, QNLI, RTE, and STS-B. Following the experimental setup of (Wu et al., 2024a,c; Bini et al., 2025), we randomly split the validation set into two subsets using a pre-defined seed: one subset for model hyperparameter selection and the other is reserved for final performance evaluation. Detailed statistics for each dataset are provided in Appendix B.1

Implementation Details. We follow the experimental setup used in (Wu et al., 2024a,c; Bini et al., 2025) to ensure fair comparison across various methods. To maintain a consistent number of trainable parameters among different approaches, we fine-tune only the query (Q) and value (V) projection layers in the attention modules and fix the rank to $r = 8$ for all datasets. ALL reported results are averaged over five runs with different random seeds. Additional implementation and hyperparameter details can be found in Appendix B.2

Main Results. Table 2 reports the NLU results on six benchmark datasets using different RoBERTa models. PERA consistently outperforms all existing PEFT methods across every evaluated configuration. Notably, under the same rank setting ($r = 8$), PERA achieves average accuracy gains of 1.70% and 0.83% over LoRA on RoBERTa-base and RoBERTa-large, respectively. Moreover, on the RoBERTa-large model, PERA delivers the best performance on all six datasets, demonstrating its strong generalization and the effectiveness of high-order nonlinear feature modeling even in

Model	Method	#Param	SST-2	MRPC	CoLA	QNLI	RTE	STS-B	Avg.
RoBERTa-base	FFT	125M	94.40	87.90	62.40	92.50	78.30	90.60	84.40
	RED	0.02M	93.90	89.20	61.00	90.70	**78.00**	90.40	83.90
	LoRA	0.3M	93.90	88.70	59.70	92.60	75.30	90.30	83.40
	HiRA	0.3M	**94.22**	89.53	60.30	92.39	74.68	89.58	83.45
	DeLoRA	0.3M	94.10	89.00	63.60	92.80	77.10	90.90	84.60
	PERA	0.3M	94.13	**89.71**	**64.80**	**92.94**	77.84	**91.13**	**85.10**
RoBERTa-large	FFT	355M	96.00	91.70	68.20	93.80	85.80	92.60	88.00
	RED	0.05M	96.00	90.30	68.10	93.50	86.20	91.30	87.60
	LoRA	0.8M	96.00	89.80	65.50	94.70	86.30	91.70	87.30
	PERA	0.8M	**96.24**	**90.78**	**68.40**	**94.89**	**86.47**	**91.98**	**88.13**

Table 2: Comparisons of different methods finetuning RoBERTa-base and RoBERTa-large on GLUE benchmark. Results of all baselines are taken from (Bini et al., 2025; Wu et al., 2024c) except HiRA. The best performance within each LLM is indicated in bold, while the second best performance is highlighted in underline.

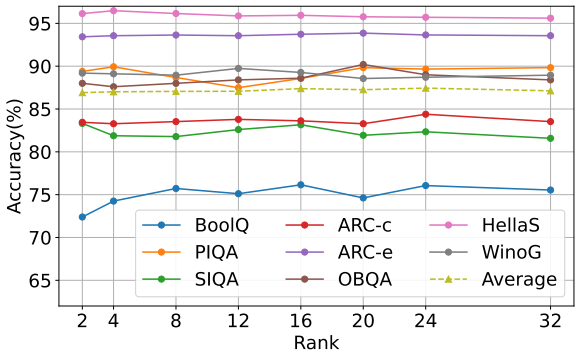


Figure 3: Accuracy(%) comparison with the rank r increases on LLaMA3-8B model. The detailed results are provided in Appendix C.1

high-capacity models.

5 Understanding the PERA

5.1 Impact on the Number of Rank

We conduct a systematic study to investigate how the parameter rank r influences model performance across multiple commonsense reasoning tasks. In the experiments, we vary only the value of r while keeping all other hyperparameters fixed and evaluate the LLaMA3-8B model using the average accuracy as the evaluation metric. The results, as shown in Figure 3, reveal several important observations: First, even under extremely low-rank settings ($r = 2$ and $r = 4$), PERA achieves strong performance, with average accuracy of 86.91% and 87.01%. This indicates that PERA can effectively leverage low-rank parameters to construct high-order interaction terms, thereby enriching adaptation representations. Second, as the rank increases from 2 to 16, performance improves substantially, with the average accuracy rising from 86.91% to 87.38%, which corresponds to the best observed result. These observations suggest that PERA can achieve performance comparable to high-resource configurations even in low-resource settings. This rank robustness can be attributed primarily to the polynomial expansion mechanism, which effectively increases the maximum achievable rank of updated weights and enables high-order interactions within the low-rank space, thereby enhancing the expressive capacity of the model.

Placement	#Params(%)	Avg.
QKV, UD	0.3513	**87.38**
QKV	0.1176	87.02
UD	0.2345	86.91
QK	0.0849	85.25
QV	0.0849	86.70
Q	0.0522	84.83
K	0.0326	83.53
V	0.0326	86.19

Table 3: Average accuracy (%) comparison of different modules on various tasks with LLaMA3-8B model.

5.2 Impact on Placement in Transformers

In this section, we analyze the effect of applying PERA to different weight modules within the Transformer architecture. Each module is denoted by its first letter: (Q)uery, (K)ey, (V)alue, (U)p, and (D)own. As shown in Table 3, the experimental results show that applying PERA jointly to both the QKV and UD layers yields the best performance, which is consistent with the structure recommended by LoRA. In contrast, restricting PERA to a single module leads to a degradation in overall perfor-

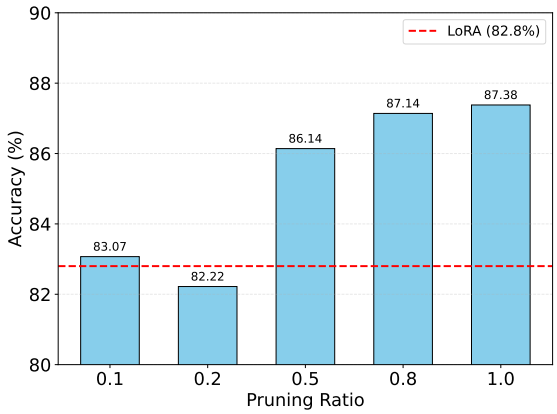


Figure 4: Average accuracy under different pruning ratios of Commonsense170K with LLaMA3-8B model.

mance. Furthermore, adapting the QKV layers consistently outperforms adapting only the UD layers. The detailed results can be found in Appendix C.2

Method	ΔW	**Avg.**
LoRA	Eq. 8	82.80
LoRA + square-only	Eq. 10	87.48
LoRA + cross-only	Eq. 11	86.83
PERA	Eq. 9	87.38

Table 4: Ablation of PERA innovations on the commonsense reasoning tasks. We show how different components affect performance from LoRA (Hu et al., 2022) with the LLaMA3-8B model.

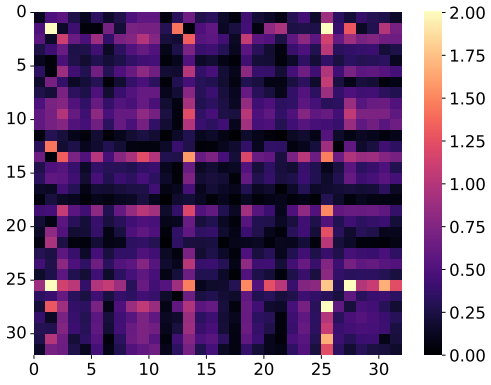
5.3 Impact on Low Resources

To evaluate the performance of PERA under low-resource conditions, we partition the commonsense170k dataset and randomly sample 10%, 20%, 50% and 80% of the data for experimentation. Figure 4 illustrates the relationship between training dataset pruning ratio and average accuracy, with detailed numerical results reported in the Appendix C.3. The results demonstrate that even when trained on only 10% of the dataset, PERA achieves an average accuracy of 83.07%, surpassing LoRA (82.80%) trained on the full commonsense170K dataset. Moreover, as the number of training datasets increases, the performance gap between PERA and LoRA widens, highlighting its data efficiency in low-resource settings.

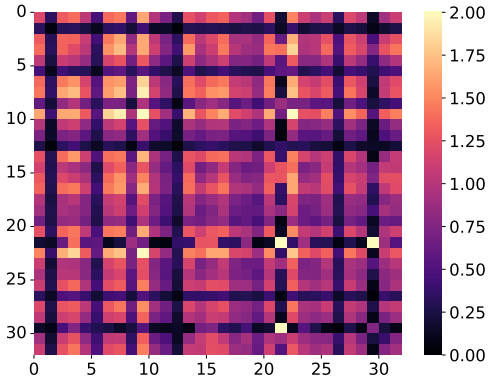
5.4 Impact on Different High-Order Components with LoRA

Since LoRA can be viewed as a special case of PERA under specific initialization, we further examine the individual effects of introducing high-order square and cross feature terms. The standard LoRA formulation is shown in Eq. 8, while the square-only and cross-only variants are defined in Eq. 10 and Eq. 11, respectively. PERA (Eq. 9) can thus be regarded as a generalized LoRA variant that jointly incorporates both types of high-order terms. The result is shown in Table 4.

Experiments on commonsense reasoning tasks using LLaMA3-8B (rank = 16) show that incorporating either square (87.48%) or cross (86.83%) terms notably improves performance over LoRA (82.80%). Among them, the square feature expansion yields the largest gain, highlighting its importance in enhancing model expressivity. Combining both high-order terms (PERA) yields a comparable improvement (87.38%), suggesting that excessive interaction modeling may introduce redundancy.



(a) LoRA.



(b) PERA.

Figure 5: The interaction strength matrix of LoRA and PERA. We quantify the combined influence of each feature dimension on the output, not just the individual influence.

5.5 Feature Interaction Strength Analysis

We compare our method with LoRA using the Hessian-based Interaction Strength Matrix. Specif-

Method	Training		Inference	
	Memory	Speed	Memory	Speed
LoRA	18.70GB	12h 27m	19.50GB	9m30s
DoRA	19.58GB	22h 07m	19.94GB	23m40s
HiRA	18.72GB	14h 02m	19.58GB	20m45s
PERA	19.12GB	13h 30m	19.70GB	14m53s

Table 5: Comparison of training and inference efficiency across methods.

ically, for the model output $f(h)$, we compute the absolute second-order partial derivatives with respect to the hidden representation:

$$S_{ij} = \mathbb{E}_h \left[\left| \frac{\partial^2 f(h)}{\partial h_i \partial h_j} \right| \right]. \tag{12}$$

As shown in Figure 5, we randomly select the slices of a small set of hidden states. Our method yields a higher global interaction strength compared to LoRA, indicating a more expressive modeling of high-order feature coupling.

5.6 Memory and Speed Analysis

To provide a more comparison on training and inference cost, we compare PERA with LoRA, DoRA, and HiRA under identical settings (Commonsense170K, rank=4, batch size=16, LLaMA2-7B, 3 epochs and evaluating on BoolQ dataset) on the same single RTX5090 32G GPU. We report peak memory usage and total training time. The results are summarized in Table 5.

PERA implements higher-order interaction terms via matrix concatenation rather than sequential matrix additions, which avoids additional forward passes and limits computational overhead. Overall, the results indicate that PERA maintains a computational and memory footprint close to standard LoRA and remain substantially more efficient than DoRA.

6 Conclusion

We introduce PERA, a polynomial expansion–based low-rank adaptation method that enables higher-order parameter interactions while preserving efficiency. Our analysis and experiments highlight the importance of high-order nonlinear components, especially square terms, in improving the expressive capacity of low-rank fine-tuning. These results indicate that structured modeling of higher-order parameter relationships offers a promising direction for more expressive and efficient adaptation.

7 Limitations

In this work, we primarily evaluate PERA on commonsense reasoning tasks and GLUE benchmarks using LLaMA and RoBERTa models. While these results demonstrate the effectiveness of PERA in language understanding and reasoning scenarios, the current evaluation does not cover other domains that may require different forms of adaptation. In particular, tasks such as arithmetic reasoning or multimodal generation (e.g., image generation) may exhibit distinct characteristics and challenges. Exploring the applicability and effectiveness of PERA in these domains remains an important direction for future work.

8 Ethics Statement

This work introduces PERA, a parameter-efficient fine-tuning method that operates entirely in the parameter space of pretrained language models. The research does not involve the collection or annotation of new data, human subjects, or user-generated content. All datasets used in our experiments are publicly available and widely adopted in prior PEFT studies. The proposed method aims to improve model expressivity and efficiency without altering model behavior or generating new textual content. Therefore, it poses no additional ethical or societal risks beyond standard fine-tuning techniques. We encourage responsible use of PEFT methods, particularly regarding dataset bias and downstream deployment contexts.

9 Acknowledgements

This work is supported by the National Natural Science Foundation of China (No.62206004, No.62572002, No.62272001, No.624065095), and the Natural Science Foundation of Anhui Province (No.2208085QF199, No.2508085MF159, No.2308085MF213).

References

Massimo Bini, Leander Girrbach, and Zeynep Akata. 2025. Delora: Decoupling angles and strength in low-rank adaptation. *arXiv preprint arXiv:2503.18225*.

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, volume 34, pages 7432–7439.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

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*.

Dorottya Demszky, Kelvin Guu, and Percy Liang. 2018. Transforming question answering datasets into natural language inference datasets. *arXiv preprint arXiv:1809.02922*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the third international workshop on paraphrasing (IWP2005)*.

Carl Eckart and Gale Young. 1936. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218.

Soledad Galli. 2020. Python feature engineering cookbook: Over 70 recipes for creating. *Engineering, and Transforming Features to Build Machine Learning Models*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

R Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, volume 7, pages 785–794.

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*.

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

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

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*.

Max Kuhn and Kjell Johnson. 2019. *Feature engineering and selection: A practical approach for predictive models*. Chapman and Hall/CRC.

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

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. 2023. Scaling down to scale up: A guide to parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.15647*.

Mingjie Liu, Teodor-Dumitru Ene, Robert Kirby, Chris Cheng, Nathaniel Pinckney, Rongjian Liang, Jonah Alben, Himyanshu Anand, Sanmitra Banerjee, Ismet Bayraktaroglu, and 1 others. 2023. Chipnemo: Domain-adapted llms for chip design. *arXiv preprint arXiv:2311.00176*.

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 Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

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*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Sinan Ozdemir. 2022. *Feature engineering bookcamp*. Simon and Schuster.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. 2023. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22500–22510.

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

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

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. 2023. Large language models in medicine. *Nature medicine*, 29(8):1930–1940.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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 Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pages 353–355.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long papers)*, pages 1112–1122.

Muling Wu, Wenhao Liu, Xiaohua Wang, Tianlong Li, Changze Lv, Zixuan Ling, Zhu JianHao, Cenyuan Zhang, Xiaoqing Zheng, and Xuan-Jing Huang. 2024a. Advancing parameter efficiency in fine-tuning via representation editing. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13445–13464.

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

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. 2024c. Reft: Representation finetuning for language models. *Advances in Neural Information Processing Systems*, 37:63908–63962.

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*.

Liu Zhuang, Lin Wayne, Shi Ya, and Zhao Jun. 2021. A robustly optimized bert pre-training approach with post-training. In *Proceedings of the 20th chinese national conference on computational linguistics*, pages 1218–1227.

Zhan Zhuang, Yulong Zhang, Xuehao Wang, Jiangang Lu, Ying Wei, and Yu Zhang. 2024. Time-varying lora: Towards effective cross-domain fine-tuning of diffusion models. *Advances in Neural Information Processing Systems*, 37:73920–73951.

A Analysis Proof

A.1 Rank Analysis

LoRA. The weight update matrix ΔW can be decomposed into the product of two low-rank matrices $A \in \mathbb{R}^{r\times n}$ and $B \in \mathbb{R}^{m\times r}$. Given a pretrained weight matrix $W_0 \in \mathbb{R}^{m\times n}$ with rank r_0, the rank of the final weight matrix W' with LoRA satisfies the following relationship:

$$\mathrm{Rank}(W') = \mathrm{Rank}(W_0 + \Delta W) = \mathrm{Rank}(W_0 + BA).$$

According to the properties of rank in a matrix, the upper bound of LoRA can be expressed as follows:

$$\begin{aligned}\mathrm{Rank}(W_0 + BA) &\leq \mathrm{Rank}(W_0) + \mathrm{Rank}(BA)\\ &\leq \mathrm{Rank}(W_0) + \min(\mathrm{Rank}(B), \mathrm{Rank}(A))\\ &\leq r_0 + r.\end{aligned}$$

PERA. PERA first applies the polynomial expansion to the low-rank matrix $B \in \mathbb{R}^{m\times r}$ to obtain a high-rank matrix $\hat{B} \in \mathbb{R}^{m\times(2r+C(r,2))}$. Meanwhile, the polynomial expansion with Hadamard product is applied to the low-rank matrix $A \in \mathbb{R}^{r\times n}$ to generate a high-rank matrix $\hat{A} \in \mathbb{R}^{(2r+C(r,2))\times n}$. The weight update matrix can be decomposed as the product of these two high-rank matrices. The rank of the final weight matrix W' with PERA satisfies the following relationship:

$$\mathrm{Rank}(W') = \mathrm{Rank}(W_0 + \Delta W) = \mathrm{Rank}\big(W_0 + \mathrm{Poly}^2(B)\,\mathrm{Poly}_H^2(A)\big)\,.$$

According to the properties of rank in a matrix, the upper bound of PERA can be expressed as follows:

$$\begin{aligned}\mathrm{Rank}\big(W_0 + \mathrm{Poly}^2(B)\,\mathrm{Poly}_H^2(A)\big) &\leq \mathrm{Rank}(W_0) + \mathrm{Rank}\big(\mathrm{Poly}^2(B)\,\mathrm{Poly}_H^2(A)\big)\\ &\leq \mathrm{Rank}(W_0) + \min\big(\mathrm{Rank}\big(\mathrm{Poly}^2(B)\big)\,, \mathrm{Rank}\big(\mathrm{Poly}_H^2(A)\big)\big)\\ &\leq r_0 + (2r + C(r,2)).\end{aligned}$$

A.2 Feature Utilization Analysis

LoRA. In LoRA, the weight update matrix is constructed by the product of two low-rank matrices $B \in \mathbb{R}^{m\times r}$ and $A \in \mathbb{R}^{r\times n}$:

$$\Delta W = BA.$$

The low-rank matrices B and A can be represented in terms of column and row vectors, respectively:

$$B = [\,\mathbf{b}_i \mid 1 \leq i \leq r\,];$$

$$A = [\,\mathbf{a}_i \mid 1 \leq i \leq r\,]^T,$$

where each $\mathbf{b}_i \in \mathbb{R}^m$ corresponds to a column vector in B and each $\mathbf{a}_i \in \mathbb{R}^n$ corresponds to a row vector in A. Therefore, the weight update matrix ΔW of LoRA can be expressed as:

$$\Delta W = BA = \sum_{i=1}^{r} \mathbf{b}_i \mathbf{a}_i^T.$$

PERA. PERA decomposes the weight update matrix into the product of two high-rank matrices, $\hat{B} \in \mathbb{R}^{m\times(2r+C(r,2))}$ and $\hat{A} \in \mathbb{R}^{(2r+C(r,2))\times n}$:

$$\Delta W = \hat{B}\hat{A} = \mathrm{Poly}^2(B)\mathrm{Poly}_{\mathrm{H}}^2(A)$$

The low-rank matrices $\hat{B}$ and $\hat{A}$ can be represented in terms of column and row vectors, respectively:

$$\mathrm{Poly}^2(B) = \left[\begin{array}{l|l} \mathbf{b}_i; & 1 \leq i \leq r; \\ \mathbf{b}_i \odot \mathbf{b}_j; & 1 \leq i = j \leq r; \\ \mathbf{b}_i \odot \mathbf{b}_j & 1 \leq i < j \leq r \end{array}\right].$$

$$\text{Poly}_{\text{H}}^2(A) = \left[\begin{array}{l|l} \mathbf{a_i}; & 1 \le i \le r; \\ h_{ij}(\mathbf{a}_i \odot \mathbf{a}_j); & 1 \le i = j \le r; \\ h_{ij}(\mathbf{a}_i \odot \mathbf{a}_j) & 1 \le i < j \le r \end{array}\right]^T.$$

Therefore, the weight update matrix ΔW of PERA can be expressed as:

$$\begin{aligned}\Delta W &= \text{Poly}^2(B)\text{Poly}_{\text{H}}^2(A) \\ &= \sum_{i=1}^{r} \mathbf{b}_i \mathbf{a}_i^T + \sum_{1 \le i = j}^{r} h_{ij}(\mathbf{b}_i \odot \mathbf{b}_j)(\mathbf{a}_i^T \odot \mathbf{a}_j^T) + \sum_{1 \le i < j}^{r} h_{ij}(\mathbf{b}_i \odot \mathbf{b}_j)(\mathbf{a}_i^T \odot \mathbf{a}_j^T).\end{aligned}$$

A.3 Proof of PERA's Expressive Power

In this section, we give the details proof of the expressive power of HiRA in comparison to LoRA. We begin by introducing the Eckart-Young-Mirsky Theorem (Eckart and Young, 1936), which provides the optimal low-rank approximation of a matrix. We will refer to this theorem as Lemma 1.

Lemma 1 (Eckart-Young-Mirsky Theorem). The best rank-r approximation of a matrix W in the spectral norm is given by the $(r+1)$-th largest singular value, i.e.,

$$\min_{\text{Rank}(\hat{W}) \le r} \|W - \hat{W}\|_2 = \sigma_{r+1}(W).$$

Theorem 1 (The Expressive Power of PERA). Let $\bar{E}$ denote the optimal parameter update. The PERA update is defined as:

$$\Delta W_{\text{PERA}} = BA + \Delta W_{\text{square}} + \Delta W_{\text{cross}},$$

Where

- $\text{Rank}(BA) \le r$,
- $\text{Rank}(\Delta W_{\text{square}}) \le r$,
- $\text{Rank}(\Delta W_{\text{cross}}) \le C(r, 2)$.

Then the following bound holds:

$$\min_{\Delta W \in S_{\text{PERA}}} \|\Delta W - \bar{E}\|_2 \le \sigma_{2r+C(r,2)+1}(\bar{E}),$$

Where

$$S_{\text{PERA}} = BA + \Delta W_{\text{square}} + \Delta W_{\text{cross}}.$$

Proof. Since

$$\text{rank}(\Delta W_{\text{PERA}}) \le \text{rank}(BA) + \text{rank}(\Delta W_{\text{square}}) + \text{rank}(\Delta W_{\text{cross}}) \le 2r + C(r, 2)$$

The PERA hypothesis space is contained in the set of matrices with rank at most $2r + C(r, 2)$. Applying Lemma 1 yields

$$\min_{\Delta W \in S_{\text{PERA}}} \|\Delta W - \bar{E}\|_2 \le \sigma_{2r+C(c,2)+1}(\bar{E}).$$

Comparison with LoRA. For LoRA,

$$\text{rank}(\Delta W_{\text{LoRA}}) \le r,$$

Which implies

$$\min_{\Delta W_{\text{LoRA}}} \|\Delta W - \bar{E}\|_2 = \sigma_{r+1}(\bar{E}).$$

Since singular values are non-increasing

$$\sigma_{2r+C(r,2)+1}(\bar{E}) \le \sigma_{r+1}(\bar{E}),$$

Which establishes that PERA admits a strictly larger approximation space in terms of rank-constrained spectral approximation.

B Experimental Details

B.1 Datasets

Commonsense170K. This dataset integrates eight widely used benchmarks for commonsense reasoning:

- BoolQ (Clark et al., 2019): This dataset consists of 15,942 naturally occurring yes/no questions collected from unprompted and unconstrained settings.
- PIQA (Bisk et al., 2020): A multiple-choice dataset focused on physical commonsense reasoning, where each example presents a question with two possible solutions.
- SIQA (Sap et al., 2019): This dataset contains multiple-choice questions designed to evaluate a model's ability to reason about the social and pragmatic implications of everyday events.
- HellaSwag (Zellers et al., 2019): A commonsense natural language inference benchmark in which models must select the most plausible ending given a contextual prompt.
- WinoGrande (Sakaguchi et al., 2021): A fill-in-the-blank task with two candidate answers, aimed at assessing commonsense reasoning and coreference resolution.
- ARC-e and ARC-c (Clark et al., 2018): The Easy and Challenge subsets of the ARC benchmark, consisting of grade-school-level multiple-choice science questions. Notably, the Challenge set includes questions that are difficult for both retrieval-based and word co-occurrence-based methods.
- OBQA (Mihaylov et al., 2018): A dataset of elementary-level multiple-choice science questions that require multi-step reasoning, integration of external commonsense knowledge, and rich text comprehension.

GLUE. The General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018) comprises a collection of datasets designed to evaluate diverse natural language understanding capabilities. These include CoLA (Warstadt et al., 2019), SST-2 (Socher et al., 2013), MRPC (Dolan and Brockett, 2005), QQP, STS-B (Cer et al., 2017), MNLI (Williams et al., 2018), QNLI (Demszky et al., 2018), and RTE (Haim et al., 2006). In our experiments, we evaluate PERA on six representative datasets from the GLUE benchmark. For fair comparison, we follow the same dataset splits as in (Wu et al., 2024c; Bini et al., 2025). Specifically, if a validation set contains more than 2,000 samples, we randomly sample 1,000 examples to form a validation subset and use the remaining samples as the test set. The sizes of the training, validation, and test sets for all datasets are reported in Table 6.

HyperParameters	**LLaMA2-7B**		**LLaMA3-8B**	
Rank r	16	32	16	32
α	16	32	16	32
Dropout	0.05			
Optimizer	AdamW			
LR	1e-4			
LR Scheduler	Linear			
Batch Size	16			
Warmup Steps	100			
Epochs	3			
Where	Q, K, V, Up, Down			

Table 6: The hyperparameters for PERA on the commonsense reasoning tasks.

B.2 Hyperparameters

Commonsense Reasoning. Table 7 presents the hyperparameter configurations used when fine-tuning the LLaMA2-7B and LLaMA3-8B models with PERA on the commonsense reasoning. To ensure fair comparisons among LoRA, DoRA, MoRA, and HiRA, we follow the experimental setup in (Huang et al., 2025) and maintain the same or comparable numbers of trainable parameters.

GLUE. Following (Bini et al., 2025), we use the new validation set to fine-tune the hyperparameters on random seed 42. Then, we select the best hyperparameters to evaluate test performance for seeds 42, 43, 44, 45, 46. For each training run, we use checkpointing to save the best training run and evaluate with that. For all experiments, we use a max sequence length of 512. For all hyperparameters, we run a small grid search. The detailed hyperparameters are reported in Table 8.

C Detailed Experimental Results

C.1 Different Rank Setting

Table 9 illustrates the impact of the parameter rank r on model performance across multiple commonsense reasoning tasks when fine-tuning the LLaMA3-8B model.

Splits Sizes	SST-2	MRPC	CoLA	QNLI	RTE	STS-B
Training Set	67K	3.7K	8.5K	105K	2.5K	5.7K
New Validation Set	436	204	522	1K	139	750
New Test Set	436	204	521	4.5K	138	750

Table 7: GLUE dataset sizes, with new validation and test splits following (Wu et al., 2024c; Bini et al., 2025) setup

Model	Hyperparameters	SST-2	MRPC	CoLA	QNLI	RTE	STS-B
RoBERTa-base	Learning Rate	1e-4	1e-3	1e-3	3e-4	5e-4	1e-3
	Batch Size	32	32	32	32	32	32
	Num. Epochs	50	50	40	30	75	80
	Dropout	0	0.25	0	0.25	0	0.25
	Warmup Ratio	6e-2					
	Maximum Sequence Length	512					
RoBERTa-large	Learning Rate	5e-4	1e-4	4e-4	3e-4	5e-4	4e-4
	Batch Size	32	32	32	32	32	32
	Num. Epochs	45	35	60	50	75	30
	Dropout	0.25	0.0	0.20	0.0	0.25	0.10
	Warmup Ratio	6e-2					
	Maximum Sequence Length	512					

Table 8: GLUE benchmark hyperparameters.

C.2 Different Placement in Transformers

Table 10 shows the effect of applying PERA to different weight modules within the Transformer architecture. Each module is denoted by its initial letter: (Q)uery, (K)ey, (V)alue, (U)p, and (D)own. We fine-tune the LLaMA3-8B model with the rank set to 16.

C.3 Low-Resource Setting

Table 11 presents the detailed results under low-resource settings for commonsense reasoning tasks. We fine-tune the LLaMA3-8B model with the rank set to 16.

C.4 Different High-Order Components

This section presents a detailed empirical analysis of how different high-order components affect LoRA. We evaluate commonsense reasoning tasks using the LLaMA3-8B model with rank fixed at $r = 16$. The results are summarized in Table 12.

Although the square-only variant achieves the highest overall average performance, a finer-grained comparison across individual benchmarks reveals that cross terms consistently yield gains on several tasks. In particular, PERA with cross terms outperforms the square-only variant on SIQA (83.16 vs. 82.34, +0.82), ARC-e (93.73 vs. 92.80, +0.93), and HellaSwag (95.94 vs. 95.83, +0.11). These datasets typically involve multi-hop or compositional reasoning, where inter-component coupling plays a more critical role.

In contrast, on datasets such as ARC-c and PIQA, the square-only formulation performs slightly better. Overall, these results suggest that cross terms provide complementary inter-component interactions that are particularly beneficial for reasoning-intensive benchmarks, while their effectiveness may vary depending on task structure and complexity.

C.5 Training Loss Analysis

To evaluate the fitting capacity and training efficiency of PERA under a comparable parameter budget, we train all baseline methods—including representative parameter-efficient fine-tuning approaches such as DoRA and HiRA—using the same rank setting ($r = 4$) and a unified set of hyperparameters. Figure 6 presents a comparison of the training loss trajectories across different methods.

At the early stage of training, all three methods exhibit rapid convergence; however, PERA shows a noticeably steeper decline, indicating stronger initial fitting dynamics. As training progresses into a more stable phase, PERA consistently achieves lower training loss than both DoRA and HiRA, maintaining the best convergence bound throughout optimization.

Ultimately, PERA reaches a final training loss of 0.0425, whereas DoRA and HiRA obtain 0.1595 and 0.0915, respectively. These results suggest that, under the same rank constraint, PERA's higher-order nonlinear structure enables it to capture more

Figure 6: Loss Comparision between Different PEFT Methods

complex feature mappings than conventional linear low-rank adaptation methods, thereby achieving superior task performance with the same parameter efficiency.

Rank	#Param (%)	BoolQ	PIQA	SIQA	ARC-c	ARC-e	OBQA	HellaS	WinoG	Avg.
2	0.0440	72.39	89.39	83.32	83.45	93.43	88.00	96.14	89.19	86.91
4	0.0880	74.25	89.93	81.88	83.28	93.56	87.60	96.48	89.11	87.01
8	0.1760	75.72	88.68	81.78	83.53	93.64	88.00	96.15	88.95	87.05
12	0.2638	75.11	87.49	82.60	83.79	93.56	88.40	95.87	89.74	87.07
16	0.3515	76.15	88.57	83.16	83.62	93.73	88.60	95.94	89.27	87.38
20	0.4391	74.62	89.83	81.93	83.28	93.86	90.20	95.77	88.56	87.25
24	0.5266	76.06	89.66	82.34	84.39	93.65	89.00	95.71	88.71	87.44
32	0.7012	75.54	89.83	81.58	83.53	93.56	88.40	95.61	88.95	87.12

Table 9: Accuracy (%) comparison with the rank r increases on LLaMA3-8B model.

Placement	#Params (%)	BoolQ	PIQA	SIQA	ARC-c	ARC-e	OBQA	HellaS	WinoG	Avg.
QKV, UD	0.3513	**76.15**	88.57	**83.16**	**83.62**	93.73	88.60	95.94	**89.27**	**87.38**
QKV	0.1176	74.71	**89.39**	82.50	**83.62**	**93.81**	87.60	95.92	88.63	87.02
UD	0.2345	72.66	89.61	82.45	83.53	93.39	**89.00**	**96.25**	88.40	86.91
QK	0.0849	71.96	88.30	80.96	80.12	93.01	86.80	95.01	85.87	85.25
QV	0.0849	74.80	89.12	81.73	83.19	93.01	87.00	96.03	88.71	86.70
Q	0.0522	71.80	88.63	80.71	79.78	92.26	84.60	94.99	85.87	84.83
K	0.0326	70.86	87.65	78.25	78.67	91.96	82.80	93.84	84.21	83.53
V	0.0326	71.93	88.63	82.60	82.42	93.52	87.60	95.69	87.13	86.19

Table 10: Accuracy (%) comparison of different placement on commonsense reasoning tasks with LLaMA3-8B model.

Pruning Ratio	BoolQ	PIQA	SIQA	ARC-c	ARC-e	OBQA	HellaS	WinoG	Average
0.1	68.78	86.67	77.38	80.46	92.26	82.60	92.03	84.37	83.07
0.2	68.84	84.93	77.79	79.44	91.75	81.60	91.04	82.40	82.22
0.5	73.52	88.41	80.91	82.94	93.22	85.60	95.30	89.19	86.14
0.8	75.60	89.06	82.04	84.56	93.77	87.60	96.19	88.32	87.14
1.0	76.15	88.57	83.16	83.62	93.73	88.60	95.94	89.27	87.38

Table 11: Accuracy (%) comparision of different training samples pruning ratio fine-tuning LLaMA3-8B model on the commonsense reasoning tasks.

Component	BoolQ	PIQA	SIQA	ARC-c	ARC-e	OBQA	HellaS	WinoG	Average
LoRA	72.30	86.70	79.30	75.70	87.70	82.80	93.50	84.80	82.80
LoRA + cross features	75.41	88.52	81.78	82.76	93.05	88.80	95.75	88.55	86.83
LoRA + square features	76.17	89.28	82.34	84.72	92.80	89.20	95.83	89.50	87.48
PERA	76.15	88.57	83.16	83.62	93.73	88.60	95.94	89.27	87.38

Table 12: Accuracy (%) comparision of different high-order components from LoRA with LLaMA3-8B model on the commonsense reasoning tasks.