An Orthogonal High-Rank Adaptation for Large Language Models

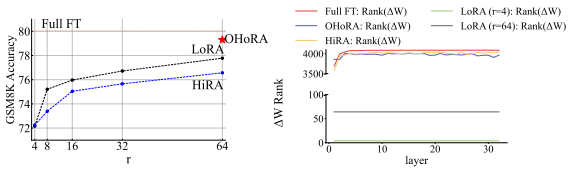
Xin Zhang[1], Guang-Ze Chen[2], Shuzhen Li[3], Zhulin Liu[1,3,*], C.L. Philip Chen[1,3], Tong Zhang[1,3]
[1]South China University of Technology, [2] University of Macau, [3]Pazhou Lab
{zx181205,chenguangze1999,melody13szli}@gmail.com,
{liuzhl,philipchen,tony}@scut.edu.cn

Abstract

Low-rank adaptation (LoRA) efficiently adapts LLMs to downstream tasks by decomposing LLMs' weight update into trainable low-rank matrices for fine-tuning. However, the random low-rank matrices may introduce massive task-irrelevant information, while their recomposed form suffers from limited representation spaces under low-rank operations. Such dense and choked adaptation in LoRA impairs the adaptation performance of LLMs on downstream tasks. To address these challenges, this paper proposes OHoRA, an orthogonal high-rank adaptation for parameter-efficient fine-tuning on LLMs. According to the information bottleneck theory, OHoRA decomposes LLMs' pre-trained weight matrices into orthogonal basis vectors via QR decomposition and splits them into two low-redundancy high-rank components to suppress task-irrelevant information. It then performs dynamic rank-elevated recomposition through Kronecker product to generate expansive task-tailored representation spaces, enabling precise LLM adaptation and enhanced generalization. OHoRA effectively operationalizes the information bottleneck theory to decompose LLMs' weight matrices into low-redundancy high-rank components and recompose them in rank-elevated manner for more task-tailored representation spaces and precise LLM adaptation. Empirical evaluation shows OHoRA's effectiveness by outperforming LoRA and its variants and achieving comparable performance to full fine-tuning with only 0.0371% trainable parameters.

1 Introduction

Fine-tuning has become a highly effective approach to align Large Language Models (LLMs) with task-specific behaviors and boost their performance on target tasks, as LLMs demonstrate strong generalization capabilities across multiple tasks (Kong et al., 2024; Liu et al., 2024a; Dong et al., 2024; Tian et al., 2024). However, with model size significantly increasing, the computational cost of fine-tuning becomes prohibitively expensive. To address these issues, Parameter efficient fine-tuning (PEFT) methods (Lester et al., 2021; Li and Liang, 2021a,b; Hu et al., 2022) have been proposed to adapt LLMs by updating only a small number of parameters. Among them, LoRA (Hu et al., 2022) has become the most widely adopted PEFT approach because of its easy deployment and strong performance comparable to full fine-tuning.



(a) (b)

Figure 1: (a) Performance of LLaMA-3.1-8B-I on GSM8K. (b) Rank comparison of ΔW among LoRA, HiRA, Full FT, and the proposed OHoRA.

However, previous studies (Tian et al., 2024; Jiang et al., 2024; Huang et al., 2025) have demonstrated that LoRA and its variants (Dettmers et al., 2023; Kopiczko et al., 2024) struggle with complex tasks. One key limitation is that the weight update ΔW is approximated by the product of two low-rank matrices, A and B, such that $\Delta W = AB$. This inherently constrains ΔW to low-rank updates, since $rank(\Delta W) \leq \min\{rank(A), rank(B)\}$, and leads to reduced expressiveness. Recent work has sought to overcome this limitation by increasing the rank of ΔW. RandLoRA (Albert et al., 2025) represented ΔW as a sum of multiple fixed random low-rank factor pairs to achieve full-rank adaptation. HiRA (Huang et al., 2025) used the frozen W_0 in LLMs as a predefined high-rank matrix and combined it with the Hadamard product to realize high-rank

*Corresponding author

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 18815–18833
November 4-9, 2025

adaptation. Although these methods successfully enrich the expressiveness of ΔW, non-trainable random matrices cannot directly interact with specific tasks and introduce task-irrelevant redundant information. The incorporation of rich prior knowledge from W_0 provides important guidance for downstream adaptation; however, not all such prior knowledge contributes positively to task-specific performance. Figure 1 further demonstrates that inclusion of W_0 achieves a high-rank update comparable to full fine-tuning, yet model performance remains inferior to that of LoRA due to the introduction of task-irrelevant redundant information. Consequently, due to the misalignment between the random matrices or model priors and target task, current static high-rank structures may introduce task-irrelevant redundant information undermining performance despite achieving high rank.

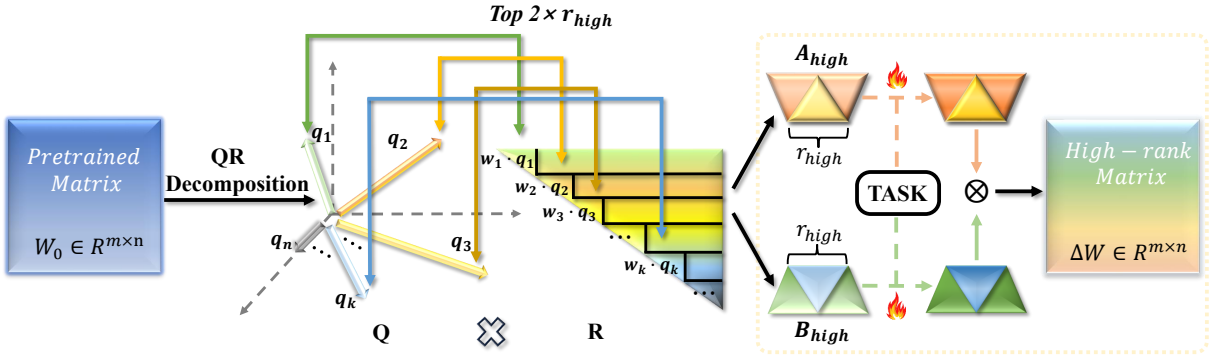
Combining the above observations, inspired by Information Bottleneck (IB) theory (Tishby et al., 2000), which explicitly decomposes input representations to remove task-irrelevant redundancy and then recomposes the most task-tailored information, this paper proposes OHoRA, an orthogonal high-rank parameter-efficient adaptation method. Specifically, given the frozen W_0, OHoRA decomposes W_0 into two low-redundancy high-rank components via an orthogonal redundancy-elimination scheme based on QR decomposition to minimize the introduction of task-irrelevant information. It then performs dynamic rank-elevated recomposition through Kronecker product to generate expansive task-tailored representation spaces, enabling precise LLM adaptation and enhanced generalization. Additionally, OHoRA optimizes the computation of Kronecker product to reduce training costs. Extensive experiments on commonsense reasoning, mathematical reasoning, code generation, cross-task NLU, and multi-turn dialogue tasks validate the effectiveness of OHoRA. In conclusion, this paper makes the following contributions:

- OHoRA pioneers an information bottleneck-driven high-rank adaptation paradigm for LLMs. It operationalizes the information bottleneck theory to decompose LLMs' weight matrices in redundancy-elimination manner and then recompose them in rank-elevated manner. It efficiently performs precise adaptation from cluttered representation spaces to expansive task-tailored ones with only 0.0371% trainable parameters.
- This paper designs an efficient optimization algorithm for Kronecker product in dynamic rank-elevated recomposition. It reduces the computational complexity from $O(mn)$ to $O(r(m+n))$ and memory usage from $O(mn)$ to $O(r^2)$.
- This paper conducts extensive experiments on various downstream tasks to validate the effectiveness of OHoRA.

2 Related Work

PEFT methods are proposed as a solution to the challenges of catastrophic forgetting, overfitting, and substantial computational overhead when adapting LLMs to downstream tasks. These methods typically freeze the original model weights and add a few additional trainable parameters to achieve adaptation and have been developed into three main categories: LoRA-based, Adapter-based and Soft Prompt-based approaches. This paper mainly focuses on LoRA-based methods. Related work on adapter-based and soft prompt-based methods is provided in Appendix C.

Low-rank adaptation LoRA (Hu et al., 2022) modeled the weight change ΔW as the product of two low-rank matrices, i.e., $\Delta W = AB$ without altering the model architecture or introducing inference delay. Following LoRA, AdaLoRA (Zhang et al., 2023) dynamically adjusted the rank in each model layer allowing fine-grained adaptation to target task. DoRA (Liu et al., 2024b) used weight normalization to decompose ΔW into a magnitude component to learn the scale of ΔW, and a direction component, which plays the same role in LoRA. OLoRA (Büyükakyüz, 2024) maintained the same architecture as LoRA and initialized two low-rank matrices leveraging orthonormal matrices. RandLoRA (Albert et al., 2025) used learned linear combinations of random low-rank matrices to perform full-rank updates. HiRA (Huang et al., 2025) applied Hadamard product to increase the rank of updated matrices and enhanced the model's expressive power. Unlike LoRA and its successors, which constrain the update of ΔW in low-rank form or rely on static high-rank structures to enhance expressiveness, OHoRA avoids the task-irrelevant redundancy from random or predefined high-rank information and uses Kronecker product to improve model expressiveness while dynamically updating high-rank matrices to better adapt to target tasks.



(a) Orthogonal Redundancy-Elimination **(b) Dynamic Rank-elevated Recomposition**

Figure 2: The overall structure of OHoRA: (a) Perform QR decomposition on W_0 and select the top $2 \times r_{high}$ non-redundant essential subspaces ranked by $w_k \cdot q_k$, where r_{high} is the largest common divisor of both m and n, constrained by $r_{high} \leq \min\{\sqrt{m}, \sqrt{n}\}$. (b) Initialize A_{high} and B_{high} from the selected essential subspaces, and use Kronecker product to achieve dynamic rank-elevated recomposition to generate a task-tailored high-rank ΔW.

Overview of Information Bottleneck (IB) theory. The IB theory seeks a compact representation T of the input X that discards those aspects of X unhelpful (task-irrelevant noise) for predicting the target Y. Formally, it optimizes

$$\min_{p(t|x)} I(X;T) - \beta I(T;Y), \tag{1}$$

where $I(X;T)$ measures how much of the original input information is retained; $I(T;Y)$ measures how well T predicts the target. By minimizing this objective, IB theory explicitly filters out task-irrelevant features and preserves only the information necessary for predictions. In PEFT paradigm, W_0 is typically kept frozen. Under this setting, the corresponding IB optimized objective in OHoRA becomes:

$$\min_{p(\Delta W|Y)} I(W_0;\Delta W) - \beta I(\Delta W;Y), \tag{2}$$

where $I(W_0;\Delta W)$ measures how much redundant information from W_0 is retained in ΔW; $I(\Delta W;Y)$ quantifies the contribution of ΔW to the target task Y.

3 Methodology

3.1 OHoRA

Existing high-rank adaptation methods introduce frozen or random high-rank matrices to compensate for the low-rank limitations of ΔW on complex tasks, thereby striking a trade-off between high-rank expressivity and parameter-efficient training. However, these matrices introduce massive task-irrelevant redundancy and degrade adaptation performance. To address this issue, this paper proposes OHoRA, an **O**rtho**g**onal **H**igh-**R**ank parameter-efficient **A**daptation for LLMs. According to the IB theory, which decomposes irrelevant redundant features to recompose more task-relevant information, OHoRA decomposes W_0 into two low-redundancy high-rank components via an orthogonal redundancy-elimination scheme based on QR decomposition. It then performs dynamic rank-elevated recomposition through Kronecker product to effectively convert high-rank information into performance gains, enabling precise LLM adaptation and enhanced generalization. The overall framework is illustrated in Figure 2. The entire algorithm is summarized in Algorithm 1.

3.2 Orthogonal Redundancy-Elimination

Although random initialization or the incorporation of model priors grants LoRA and its variants good generalization (Meng et al., 2024; Wang et al., 2024), the resulting randomness or the excessive injection of prior knowledge also introduces enormous task-irrelevant redundancy, which can lead to slow convergence and suboptimal performance (He et al., 2025; Jiang et al., 2025). To address these issues, this paper proposes an orthogonal redundancy-elimination method to decompose W_0 into two low-redundancy high-rank components to suppress task-irrelevant information, inspired by the IB theory that emphasizes the decomposition of redundant information to retain high-quality representations (i.e., minimizing $I(W_0;\Delta W)$ in equation (2)). This approach analyzes the redundancy in W_0 through orthogonal projection derived from QR decomposition, and decomposes W_0 into two high-rank components that selectively retain subspaces with significant impact on parameter up-

dates.

The QR decomposition decomposes a matrix into an orthogonal factor and a triangular factor and can be achieved by Gram-Schmidt orthogonalization. Let $W = [w_1, \ldots, w_n] \in \mathbf{R}^{m\times n}$; the Gram-Schmidt orthogonalization produces orthogonal factor $Q = [q_1, \ldots, q_n] \in \mathbf{R}^{m\times n}$ and triangular factor $R \in \mathbf{R}^{n\times n}$ as follows:

$$W = [q_1, \cdots, q_n] \begin{bmatrix} w_1 \cdot q_1 & \cdots & w_n \cdot q_1 \\ 0 & \cdots & w_n \cdot q_2 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & w_n \cdot q_n \end{bmatrix} = QR. \tag{3}$$

And each diagonal element of R can be written as:

$$R_{kk} = w_k \cdot q_k = \|w_k - \sum_{i=1}^{k-1}(w_k \cdot q_i)q_i\|, \tag{4}$$

where $\sum_{i=1}^{k-1}(w_k \cdot q_i)q_i$ can be considered as the redundant information in w_k within $q_1, q_2, ..., q_k$, the detailed procedure is provided in Appendix A. Therefore, R_{kk} quantifies the magnitude of the non-redundant information in w_k after removing its projections onto the orthogonal basis vectors $q_1, q_2, ..., q_k$. A larger value of R_{kk} indicates less redundancy and contributes more significantly to the weight update. In this way, OHoRA ranks and filters all knowledge in the pretrained model based on a "non-redundancy" metric R_{kk}, and the two high-rank components $A_{high} \in \mathbf{R}^{\frac{m}{r_{high}}\times r_{high}}$ and $B_{high} \in \mathbf{R}^{\frac{n}{r_{high}}\times r_{high}}$ can be constructed according to $w_k \cdot q_k$ as follows: Let $\lambda = [w_1 \cdot q_1, ..., w_n \cdot q_n]$ and π be a permutation of $\{1, 2, ..., n\}$ such that

$$\lambda_{\pi(1)} \geq \lambda_{\pi(2)} \geq \cdots \lambda_{\pi(n)}. \tag{5}$$

Defines the index sets as:

$$\mathcal{I}_1 = \{\pi(i) | 1 \leq i \leq r_{high}\}, \tag{6}$$

$$\mathcal{I}_2 = \{\pi(i) | r_{high} < i \leq 2r_{high}\}, \tag{7}$$

where $r_{high} = \max\{k \in \mathbb{N}^+ \mid k \leq \min\{\sqrt{m}, \sqrt{n}\}, m \bmod k = 0, n \bmod k = 0\}$. It maximizes the retention of critical information, and enforces consistency between the dimensionality of the Kronecker product of A_{high} and B_{high} and that of ΔW. Let $Q_1 = Q_{:,\mathcal{I}_1}, R_1 = R_{\mathcal{I}_1,:}$ and $Q_2 = Q_{:,\mathcal{I}_2}, R_2 = R_{\mathcal{I}_2,:}$. Then, the low-redundancy components are defined as:

$$A_{high} = Q_{1[:m/r_{high}]} R_{1[:,:r_{high}]}, \tag{8}$$

$$B_{high} = Q_{2[:n/r_{high}]} R_{2[:,:r_{high}]}. \tag{9}$$

After decomposing the essential update information of W_0 into high-rank components, OHoRA stores the remaining task-irrelevant information in frozen W_{res} following (Meng et al., 2024). It preserves the model's foundational capabilities and avoids the alteration of initial model behavior due to non-zero initialization:

$$W_{res} = W_0 - A_{high} \otimes B_{high}^\top. \tag{10}$$

The corresponding forward process is presented as follows:

$$y = W_{res}x + (A_{high} \otimes B_{high}^\top)x. \tag{11}$$

3.3 Dynamic Rank-elevated Recomposition

Though (Huang et al., 2025; Albert et al., 2025) employ high-rank matrices to mitigate the low-rank constraints inherent in LoRA, the misalignment between the target task and non-adaptive high-rank structures may introduce task-irrelevant redundancy and degrade performance. To overcome this limitation, this paper proposes a dynamic rank-elevated recomposition method that transforms A_{high} and B_{high} into a expansive task-tailored representation spaces to enable precise LLM adaptation and enhanced generalization. Inspired by the IB principle of dynamically recomposing the most task-relevant information by updating decomposed critical representations (i.e., maximizing $I(\Delta W; Y)$ in equation (2)), this method allows dynamic adjustment of high-rank components guided by the target task. It then utilizes the rank-multiplicative property of the Kronecker product to effectively convert high-rank information into performance gains.

Given A_{high} with entries a_{ij} and B_{high}, their Kronecker product is defined as:

$$A_{high} \otimes B_{high}^\top = \underbrace{\begin{pmatrix} a_{11}B_{high}^\top & \cdots & a_{1n}B_{high}^\top \\ \vdots & \ddots & \vdots \\ a_{m1}B_{high}^\top & \cdots & a_{mn}B_{high}^\top \end{pmatrix}}_{m\times n}. \tag{12}$$

As shown in (12), the Kronecker product expands the selected core subspaces into $m \times n$ dynamic subspaces. These dynamic subspaces eventually constitute the expansive task-customized representation spaces, enabling the model to handle more complex tasks. Therefore, the Kronecker product can result in a higher rank ΔW compared to that of LoRA:

$$\begin{aligned} rank(\Delta W_{kp}) &= rank(A_{high} \otimes B_{high}^\top) \\ &= rank(A_{high}) \times rank(B_{high}^\top) \\ &> rank(\Delta W_{LoRA}) \end{aligned} \tag{13}$$

Given the forward process as shown in (11), we analyze the gradients of the two high-rank components to elucidate how OHoRA effectively leverages critical model priors to achieve dynamic high-rank recomposition. For brevity, consider a linear

neural network employing the mean squared error loss $\mathcal{L}$, and the gradients of the corresponding high-rank components are given by:

$$\frac{\partial \mathcal{L}}{\partial A_{high}} = (y - y_{true}) \left(I_n \otimes B_{\text{high}}^{\top} Z\right), \tag{14}$$

$$\frac{\partial \mathcal{L}}{\partial B_{high}} = (y - y_{true}) \left(A_{high}\, Z^{\top} \otimes I_n\right). \tag{15}$$

where $I_n \in \mathbf{R}^{n \times n}$ denotes the identity matrix, and $Z \in \mathbf{R}^{\frac{n}{r_{high}} \times r_{high}} = reshape(x)$. Expressions (14), (15) reveal that the two high-rank components, initialized with the low-redundancy parameter subspaces that dominate model updates, can swiftly adapt to target task based on performance signals, and accelerate model convergence. Furthermore, continuously updating the high-rank components can effectively transform high-rank information into task-tailored information and enhance model performance on complex tasks. Finally, we adopt the identity in (Massachusetts Institute of Technology, 2023) to avoid the prohibitive computational overhead incurred by Kronecker product:

$$(A_{high} \otimes B_{high}^{\top})x = vec(B_{high}^{\top} Z A_{high}^{\top}). \tag{16}$$

3.4 Complexity Analysis

Parameter Efficiency Analysis. Given a weight matrix $W \in \mathbf{R}^{m \times n}$, $A_{high} \in \mathbf{R}^{\frac{m}{r_{high}} \times r_{high}}$, and $B_{high} \in \mathbf{R}^{\frac{n}{r_{high}} \times r_{high}}$, the total number of trainable parameters in OHoRA is $\frac{m}{r_{high}} \times r_{high} + \frac{n}{r_{high}} \times r_{high} = m + n$. Consequently, unlike a LoRA module which requires $(m+n)r$ trainable parameters for a user-specified rank r, OHoRA maintains a fixed parameter count of $m + n$. Furthermore, OHoRA inherently preserves high-rank capability from the pre-trained model and avoids the need for tuning the rank hyperparameter r.

Optimization of Kronecker Product. Let $x \in \mathbf{R}^{n \times 1}$ be the input vector, and let $A_{high} \in \mathbf{R}^{\frac{m}{r_{high}} \times r_{high}}$ and $B_{high} \in \mathbf{R}^{\frac{n}{r_{high}} \times r_{high}}$ be the factor matrices. The calculation of $(A_{high} \otimes B_{high}^{\top})x$ can be decomposed in two steps for execution:

1) Form $C = A_{high} \otimes B_{high}^{T} \in \mathbf{R}^{m \times n}$. This requires $O(mn)$ time and $O(mn)$ memory.
2) Compute Cx. This also costs $O(mn)$ time.

Using the vectorized formulation in equation (16) to substitute the original Kronecker product, the computation then decomposes into:

1) $S = B_{high}^{T} Z \in \mathbf{R}^{r_{high} \times r_{high}}$. This requires $O(nr_{high})$ time and $O(r_{high}^2)$ memory.
2) $SA_{high}^{\top}$. This requires $O(mr_{high})$ time.

This vectorized formulation reduces computational complexity from $O(mn)$ to $O(r_{high}(m+n))$ and memory usage from $O(mn)$ to $O(r_{high}^2)$, where $r_{high} << \{m, n\}$. In practice, OHoRA replaces the original Kronecker product implementation with the vectorized formulation to reduce training time by 33% and memory usage by 27%. More details are provided in section 5.5.

4 Experiments

In this section, we evaluate the performance of OHoRA on various tasks. Initially, we use Commonsense Reasoning (Hu et al., 2023) to assess Natural Language Understanding (NLU) abilities. Subsequently, we evaluate dialogue, mathematical reasoning, coding, and multi-task NLU capabilities on different LLMs.

4.1 Baselines

We compare OHoRA with several baselines to demonstrate its effectiveness. These baselines include vanilla LoRA (Hu et al., 2022), LoRA variants with original structure: OLoRA (Büyükakyüz, 2024), Pissa (Meng et al., 2024), and LoRA variants with modified structures: DoRA (Liu et al., 2024b), HiRA (Huang et al., 2025). More details are provided in Appendix F.

4.2 Implementation Details.

All the experiments are conducted on NVIDIA 80G A100 GPUs. Following (Huang et al., 2025), LLMs are fine-tuned for 1 epoch or 3 epochs, and the AdamW optimizer (Loshchilov and Hutter, 2019) is employed with a learning rate of 0.0002. All the LoRA-relevant baselines are placed on the query, key, value weights, two linear layers (i.e., down and up projection) in attention modules, and two linear layers in FFN (i.e., gate and out projection). Each experiment is conducted with 3 different random seeds and the average performance is reported. More details about the hyperparameter setups are provided in Appendix E.

4.3 Experiments on Commonsense Reasoning

Models and Datasets. We fine-tune the LLaMA-2-7B (Touvron et al., 2023) and LLaMA-3-8B (Dubey et al., 2024) on a dataset comprising 170K training samples and evaluate the fine-tuned models on 8 commonsense reasoning benchmarks including BoolQ (Clark et al., 2019), PIQA (Bisk et al.,

Model	Method	#Parmas(%)	BoolQ	PIQA	SIQA	ARC-c	ARC-e	OBQA	HellaS	WinoG	**AVG ↑**
ChatGPT †	/	/	73.10	85.40	68.50	78.50	66.10	89.80	79.90	74.80	77.01
LLaMA-2-7B	LoRA	0.5898	69.80	79.90	79.50	64.70	79.80	81.00	83.60	82.60	77.61
	DoRA	0.6097	71.80	83.70	76.00	68.20	83.70	82.40	89.10	82.60	79.69
	HiRA	0.5898	71.22	83.35	79.53	**73.81**	86.74	84.60	88.12	83.98	81.42
	OLoRA	0.5898	69.57	84.33	**81.27**	71.08	86.03	**85.00**	93.10	83.27	81.71
	PiSSA	0.5898	68.29	82.10	78.76	69.11	85.35	77.60	92.68	80.82	79.34
	OHoRA	0.0371	**72.26**	**84.60**	80.66	72.78	**87.12**	82.80	**94.23**	**85.40**	**82.48**
LLaMA-3-8B	LoRA	0.5196	70.80	85.20	79.90	71.20	84.20	79.00	91.70	84.30	80.79
	DoRA	0.5366	74.60	89.30	79.90	80.40	90.50	85.80	95.50	85.60	85.20
	HiRA	0.5196	75.40	89.70	81.15	82.90	**93.27**	88.32	95.36	87.70	86.72
	OLoRA	0.5196	72.69	87.76	81.42	79.52	91.25	86.20	81.08	86.74	83.33
	PiSSA	0.5196	73.03	85.64	81.88	81.83	91.54	86.80	95.62	87.69	85.50
	OHoRA	0.0326	**75.96**	**89.99**	**82.14**	**83.11**	93.14	**88.80**	**96.30**	**89.11**	**87.32**

Table 1: Accuracy comparison of different PEFT methods on commonsense reasoning datasets. The symbol † indicates that the results are taken from (Huang et al., 2025).

Model	Method	#Parmas(%)	MT-Bench	MMLU	GSM8K	HumanEval		**AVG ↑**
						pass@1	pass@10	
LLaMA-2-7B	LoRA	0.5898	5.20	43.67	52.84	**19.57**	28.05	29.91
	DoRA	0.6097	5.90	44.09	52.84	18.96	29.27	30.27
	HiRA	0.5898	5.67	43.06	46.85	15.30	24.39	27.05
	OLoRA	0.5898	5.78	39.50	51.18	19.21	29.27	28.99
	PiSSA	0.5898	6.16	44.02	53.07	15.79	26.83	29.17
	OHoRA	0.0371	**6.20**	**44.25**	**60.27**	18.35	**30.49**	**31.91**
Gemma-7B	LoRA	0.5823	6.67	25.27	74.00	38.66	65.85	42.09
	DoRA	0.6042	7.10	25.27	72.02	40.49	68.29	42.63
	HiRA	0.5823	6.46	25.35	59.59	35.43	64.63	38.29
	OLoRA	0.5823	6.73	25.13	63.68	36.52	62.80	38.97
	PiSSA	0.5823	6.64	24.74	67.25	**41.16**	66.46	41.25
	OHoRA	0.0366	**7.25**	**25.42**	**74.37**	39.51	**71.34**	**43.58**
LLaMA-3.1-8B-I	LoRA	0.5196	7.46	64.49	75.97	44.70	65.85	51.69
	DoRA	0.5366	7.33	64.57	76.73	44.45	64.63	51.54
	HiRA	0.5196	**7.81**	65.13	75.04	45.79	61.59	51.07
	OLoRA	0.5196	7.25	51.97	68.16	41.10	60.37	45.77
	PiSSA	0.5196	7.30	63.65	76.80	43.84	**67.07**	51.73
	OHoRA	0.0326	7.74	**65.30**	**79.30**	**46.77**	**67.07**	**53.24**

Table 2: Comparing different PEFT methods on dialogue, multi-task nlu, math, and code benchmarks.

2020), SIQA (Sap et al., 2019), ARC-c and ARC-e (Bhakthavatsalam et al., 2021), OBQA (Mihaylov et al., 2018), HellaSwag (Zellers et al., 2019) and WinoGrande (Sakaguchi et al., 2020). The accuracy is chosen as the primary metric.

Results. As shown in Table 1, OHoRA consistently outperforms all the baseline methods in terms of average accuracy across both the LLaMA-2 7B and LLaMA-3 8B. OHoRA achieves strong performance with an exceptionally low parameter budget—0.0371% for LLaMA-2 7B and 0.0326% for LLaMA-3 8B by leveraging the Kronecker product to amplify the effective information of essential high-rank components. These findings indicate that dynamic updates to high-rank information based on the target task confer more specialized adaptation and better generalization. Using the static high-rank structures from HiRA cannot achieve as much flexibility as OHoRA does.

4.4 Experiments on Chat, Math, Code, and Multi-task NLU

Models and Datasets. To further investigate the effectiveness of OHoRA, we fine-tune the LLaMA-2-7B, Gemma-7B (Mesnard et al., 2024), and LLaMA-3.1-8B instruction (LLaMA-3.1-8B-I) on four tasks: *chat*, *math*, *code*, and *multi-task NLU*. More details are provided in Appendix E.1.

Results. OHoRA outperforms baseline methods on nearly all tasks and achieves the highest average scores across all models and tasks, as shown in Table 2. For instance, OHoRA demonstrates superior performance on GSM8K and HumanEval and highlights its effectiveness in handling complex

METHOD	BOOLQ	PIQA	SIQA	ARC-C	ARC-E	OBQA	HELLAS	WINOG	**AVG ↑**
OHORA	**75.96**	**89.99**	**82.14**	**83.11**	**93.14**	**88.80**	**96.30**	**89.11**	**87.32**
$Rand\ 2r_{high}$	50.43	89.50	80.91	80.38	92.59	86.20	94.98	86.03	82.63
$Kaiming\text{-}Init$	55.17	77.04	52.35	67.66	75.84	64.00	94.36	77.35	70.47
$w/o\ Kronecker$	73.82	86.32	81.95	82.30	92.13	87.73	95.16	87.82	85.90

Table 3: Internal Module Analysis of OHoRA. $Rand\ 2r_{high}$ denotes random selection of $2r_{high}$ diagonal elements $w_k \cdot q_k$ and uses their corresponding entries in matrices Q and R to form the respective high-rank components A_{high} and B_{high}. $Kaiming\text{-}Init$ denotes the random initialization of A_{high} and B_{high} using the default initialization method in LoRA. $w/o\ Kronecker$ indicates that A_{high} and B_{high} are initialized based on $w_k \cdot q_k$ without altering the LoRA architecture.

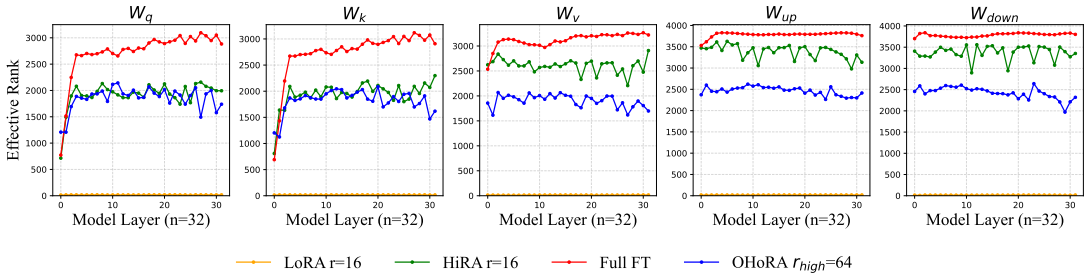


Figure 3: Effective rank across layers for Full FT, OHoRA, HiRA, and LoRA.

and diverse tasks. In contrast, although HiRA exhibits strong performance on HumanEval with the LLaMA-3.1-8B-I, it falls short of baseline methods on other models. This indicates that introducing redundant task-irrelevant high-rank information may degrade performance. OHoRA facilitates effective adaptation across various models and tasks through task-irrelevant redundancy filtering and dynamic high-rank updates.

5 Discussion and Analysis

This section analyzes the effectiveness of orthogonal redundancy-elimination and dynamic rank-elevated recomposition in OHoRA, the effective-rank comparison across layers, the impact of varying training budgets on model performance, the optimization of the Kronecker Product and the convergence behavior of OHoRA. Additional analyses can be found in Appendix G.

5.1 The effectiveness of orthogonal redundancy-elimination and dynamic rank-elevated recomposition in OHoRA

To analyze the effectiveness of orthogonal redundancy filtering and dynamic high-rank reconstruction in OHoRA, the orthogonal redundancy-elimination scheme is replaced with random initialization first. Additionally, instead of selecting the top $2r_{high}$ entries based on $w_k \cdot q_k$, $2r_{high}$ diagonal elements are randomly chosen to verify whether the two high-rank components effectively avoid the introduction of task-irrelevant redundant information. Furthermore, the Kronecker product is substituted with matrix multiplication to evaluate the impact of dynamic rank-elevated recomposition. Experiments are conducted on commonsense reasoning tasks by fine-tuning LLaMA-3-8B.

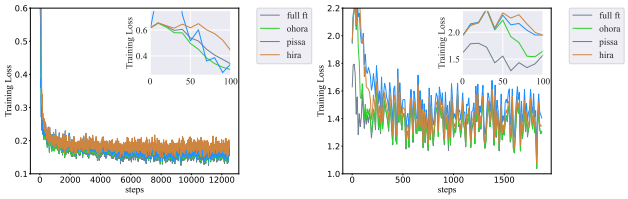
As shown in Table 8, randomly initialized high-rank components introduce substantial task-irrelevant redundancy and lead to a sharp decline in model performance. Although randomly selecting subspaces from Q and R to form A_{high} and B_{high} incorporates some useful priors and outperforms random initialization, the lack of redundancy elimination in W_0 leads to markedly inferior performance compared to OHoRA. The observed performance drop upon removing the Kronecker product shows that updating high-rank information on the fly helps the model handle complex tasks better. Section 5.2 provides the effective rank analysis to substantiate the benefits of dynamic rank-elevated recomposition. Section 5.3 further investigates the effectiveness of OHoRA components from a training dynamics perspective.

5.2 Effective Rank Comparison across Layers

To further investigate the advantage of dynamic rank-elevated recomposition, we analyze the effective rank (Roy and Vetterli, 2007) of different methods to demonstrate the effective high expressiveness of OHoRA. Effective rank measures the effective number of significant singular values by evaluating the statistical dispersion of the normalized spectrum. The larger the effective rank, the more expressive the matrix (Huang et al., 2025; Shuttleworth et al., 2024). Given a $M \times N$ matrix A, the effective rank is defined as follows:

$$erank(A) = exp\big(-\sum_{k=1}^{Q} p_k \log p_k\big), p_k = \frac{\sigma_k}{||\sigma||_1}, \quad (17)$$

where $Q = \min\{M, N\}$; σ_k is the k-th largest singular value and $\|\cdot\|_1$ is the l_1-norm of the singular values. We compare the OHoRA effective rank of ΔW with full fine-tuning (Full FT), LoRA, and HiRA across layers. In Figure 3, the largest effective rank is obtained by Full FT, which shows its high expressiveness but may also increase the risk of overfitting because of the excessive trainable parameters. The larger effective rank of HiRA comes from the pre-trained weights W_0 and the fact that W_0 is frozen during adaptation and introduces massive task-irrelevant information. OHoRA effectively mitigates the risk of overfitting and enables dynamic updates to low-redundancy high-rank components to recompose a task-tailored high-rank matrix with only a small number of trainable parameters. It improves performance and enhances generalization on complex tasks.



(a) (b)

Figure 4: (a) Training loss on MetaMathQA. (b) Training loss on Dolly-15K.

5.3 Convergence Speed of OHoRA

To analyze the dynamic training characteristics of OHoRA, its convergence speed during fine-tuning is compared with Full FT, HiRA, and Pissa. The loss curves are reported for fine-tuning LLaMA-2 7B on Dolly-15K and LLaMA-3.1-8B-I on MetaMathQA. As shown in Figure 4, OHoRA rapidly

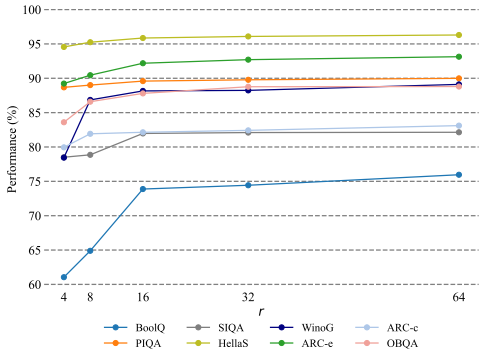


Figure 5: Performance of different rank for OHoRA.

reduces the loss to a level comparable with Full FT during the early stages of training. This indicates that OHoRA effectively filters out task-irrelevant redundancy from W_0 through orthogonal redundancy-elimination and dynamically recomposes the task-tailored high-rank matrix to accelerate adaptation to downstream tasks. In contrast, the neglect of high-rank structure in Pissa and the introduction of task-irrelevant redundancy in HiRA constrain further learning and result in slower convergence compared to OHoRA.

5.4 Impact of varying training budgets

To study the impact of rank configurations in OHoRA on model performance, numerous commonsense reasoning tasks are conducted on LLaMA-3 8B using varying rank dimensions. As illustrated in Figure 5, increasing the rank helps the model to generalize across diverse tasks, with average accuracy rising from 81.75% to 87.32% as the rank increases from 4 to 64. These results further indicate that a low-redundancy dynamic higher rank can enhance the model's capacity for complex reasoning tasks.

5.5 Optimization of Kronecker Product

To evaluate whether replacing the original Kronecker product implementation with a vectorized formulation can reduce training overhead in practice, experiments are conducted on commonsense reasoning tasks by fine-tuning LLaMA-2-7B. As illustrated in Figure 6, the optimized Kronecker product (O-Kron) achieves a significant reduction in training cost compared to the original Kronecker product (R-Kron), with approximately 1.38× lower memory usage and a 1.49× increase in training speed. Figure 7 presents an analysis of the memory and time consumption of other fine-tuning methods and highlights the training efficiency advantage of

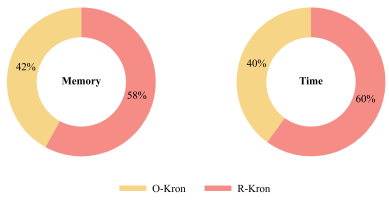


Figure 6: Memory usage and training time comparision between O-Kron and R-Kron.

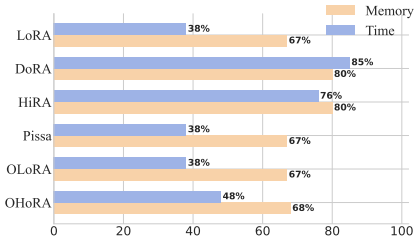


Figure 7: Training Cost Analysis. Memory represents the proportion of 80GB memory consumed by corresponding method, and Time indicates the proportion of 24 hours used for training.

OHoRA. Table 9 demonstrates that O-Kron maintains comparable performance to R-Kron.

5.6 The Efficiency of QR Decomposition

To demonstrate the efficiency and applicability of QR decomposition, two other commonly used matrix decomposition methods—SVD and LU—are compared with it in terms of computational cost, based on both theoretical analysis and numerical evaluation.

Theoretical analysis. For any matrix $A \in \mathbf{R}^{m\times n}$, the computational complexity of these methods is shown in Table 4. From the theoretical analysis shown above, both QR and SVD have a time complexity of $O(mn^2)$. However, the SVD typically involves iterative procedures, leading to higher computational cost than QR in practice.

Numerical Evaluation. To better compare the computational cost of these decomposition techniques, a matrix $W \in \mathbf{R}^{4096\times 4096}$ is applied to each method, and the FLOPs are used as the evaluation metric. The experimental results are shown in Table 5.

As shown in the Table 5, although LU decomposition is efficient, its applicability is limited to square matrices, and large-scale models rarely meet this requirement. In contrast, QR decomposition not only offers broad applicability to non-square matrices, but also maintains relatively low computational overhead, making it a practical and efficient choice for large model scenarios.

Complexity	**QR**	**SVD**	**LU**
Time	$2mn^2 - \frac{2}{3}n^3$	$4mn^2 + 8n^3$	$\frac{2}{3}n^3$

Table 4: Computational complexity comparison of different matrix decomposition methods.

METHOD	FLOPS (GFLOPS)
SVD	824.634
QR	**91.626**
LU	**45.813**

Table 5: Computational cost comparison among SVD, QR, and LU.

6 Conclusion

This paper introduces OHoRA, a method inspired by the IB theory that enables low-redundancy dynamic high-rank adaptation to enhance model performance on complex and diverse tasks. OHoRA applies orthogonal redundancy-elimination to decompose W_0 into orthogonal basis vectors via QR decomposition and splits them into two low-redundancy high-rank components to suppress task-irrelevant information. It then performs dynamic rank-elevated recomposition through Kronecker product to generate expansive task-tailored representation spaces, enabling precise LLM adaptation and enhanced generalization. Optimization for Kronecker product implementation further reduces training costs in practice and ensures efficiency alongside high-rank representation. For future work, we are interested in applying OHoRA integrated with reinforcement learning to further enhance its adaptability.

Limitations

The present study focuses on commonsense reasoning, multi-task natural language understanding, dialogue, mathematical reasoning, and code ability with LLMs. We are interested in applying OHoRA in more applications for future work. In constructing A_{high} and B_{high} from low-redundancy orthogonal subspaces, OHoRA directly selects the top n/r_{high} or m/r_{high} rows and the top r_{high} columns for each matrix. More fine-grained selection strategies may further improve the expressiveness of A_{high} and B_{high}. We leave the exploration

of such methods for future investigation.

Acknowledgments

This work was funded in part by the National Natural Science Foundation of China grant under number 62222603, in part by the STI2030-Major Projects grant from the Ministry of Science and Technology of the People's Republic of China under number 2021ZD0200700, in part by the Key-Area Research and Development Program of Guangdong Province under number 2023B0303030001, and in part by the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (2019ZT08X214).

References

Paul Albert, Frederic Z. Zhang, Hemanth Saratchandran, Cristian Rodriguez-Opazo, Anton van den Hengel, and Ehsan Abbasnejad. 2025. RandloRA: Full rank parameter-efficient fine-tuning of large models. In *The Thirteenth International Conference on Learning Representations*.

Sumithra Bhakthavatsalam, Daniel Khashabi, Tushar Khot, Bhavana Dalvi Mishra, Kyle Richardson, Ashish Sabharwal, Carissa Schoenick, Oyvind Tafjord, and Peter Clark. 2021. Think you have solved direct-answer question answering? try arc-da, the direct-answer AI2 reasoning challenge. *CoRR*, abs/2102.03315.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press.

Kerim Büyükakyüz. 2024. Olora: Orthonormal low-rank adaptation of large language models. *CoRR*, abs/2406.01775.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

Joon-Young Choi, Junho Kim, Jun-Hyung Park, Wing-Lam Mok, and SangKeun Lee. 2023. SMoP: Towards efficient and effective prompt tuning with sparse mixture-of-prompts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14306–14316, Singapore. Association for Computational Linguistics.

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. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2924–2936. Association for Computational Linguistics.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Wei Dong, Yuan Sun, Yiting Yang, Xing Zhang, Zhijun Lin, Qingsen Yan, Haokui Zhang, Peng Wang, Yang Yang, and Hengtao Shen. 2024. Efficient adaptation of pre-trained vision transformer via householder transformation. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien

Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Zhiwei He, Zhaopeng Tu, Xing Wang, Xingyu Chen, Zhijie Wang, Jiahao Xu, Tian Liang, Wenxiang Jiao, Zhuosheng Zhang, and Rui Wang. 2025. RaSA: Rank-sharing low-rank adaptation. In *The Thirteenth International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

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. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 5254–5276. Association for Computational Linguistics.

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*.

Shuyang Jiang, Yusheng Liao, Ya Zhang, Yanfeng Wang, and Yu Wang. 2025. Fine-tuning with reserved majority for noise reduction. In *The Thirteenth International Conference on Learning Representations*.

Ting Jiang, Shaohan Huang, Shengyue Luo, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, Qi Zhang, Deqing Wang, and Fuzhen Zhuang. 2024. Mora: High-rank updating for parameter-efficient fine-tuning. *CoRR*, abs/2405.12130.

Xiaoyu Kong, Jiancan Wu, An Zhang, Leheng Sheng, Hui Lin, Xiang Wang, and Xiangnan He. 2024. Customizing language models with instance-wise lora for sequential recommendation. In *Advances in Neural Information Processing Systems*, volume 37, pages 113072–113095. Curran Associates, Inc.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. 2024. Vera: Vector-based random matrix adaptation. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3045–3059. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021a. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021b. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics.

Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. 2024a. When MOE meets llms: Parameter efficient fine-tuning for multi-task medical applications. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 1104–1114. ACM.

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

2024, Vienna, Austria, July 21-27, 2024. OpenReview.net.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *CoRR*, abs/2110.07602.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 1022–1035.

Massachusetts Institute of Technology. 2023. Matrix calculus (for machine learning and beyond). MIT OpenCourseWare. [Course].

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. Pissa: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Cristian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, and et al. 2024. Gemma: Open models based on gemini research and technology. *CoRR*, abs/2403.08295.

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. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Olivier Roy and Martin Vetterli. 2007. The effective rank: A measure of effective dimensionality. In *15th European Signal Processing Conference, EUSIPCO 2007, Poznan, Poland, September 3-7, 2007*, pages 606–610. IEEE.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8732–8740. AAAI Press.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 4462–4472. Association for Computational Linguistics.

Reece Shuttleworth, Jacob Andreas, Antonio Torralba, and Pratyusha Sharma. 2024. Lora vs full fine-tuning: An illusion of equivalence. *CoRR*, abs/2410.21228.

Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Chengzhong Xu. 2024. Hydralora: An asymmetric lora architecture for efficient fine-tuning. In *Advances in Neural Information Processing Systems*, volume 37, pages 9565–9584. Curran Associates, Inc.

Naftali Tishby, Fernando C. N. Pereira, and William Bialek. 2000. The information bottleneck method. *CoRR*, physics/0004057.

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, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Shaowen Wang, Linxi Yu, and Jian Li. 2024. Lora-ga: Low-rank adaptation with gradient approximation. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. 2024. Opencodeinterpreter: Integrating code generation with execution and refinement. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 12834–12859. Association for Computational Linguistics.

A QR decomposition through Gram-Schmidt orthogonalization

The QR decomposition (also called the QR factorization) of a matrix is a decomposition of the matrix into an orthogonal matrix and a triangular matrix. The QR decomposition of a real square matrix A is as

$$A = QR, \tag{18}$$

where Q is an orthogonal matrix (i.e. $Q^TQ = I$) and R is an upper triangular matrix. There are several methods for actually computing the QR decomposition. One of such methods is the Gram-Schmidt process. Let $W \in \mathbf{R}^{m\times n}, m \geq n = \mathrm{rank}(A)$. The Gram-Schmidt orthogonalization produces $Q \in \mathbf{R}^{m\times n}$ and $R \in \mathbf{R}^{n\times n}$ in the factorization

$$W = (\mathbf{w}_1, \ldots, \mathbf{w}_n) = QR, \quad Q = (\mathbf{q}_1, \ldots, \mathbf{q}_n), \tag{19}$$

where Q has orthogonal columns and R is upper triangular. The orthonormal columns $\{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_n\}$ of Q are obtained by successively orthogonalizing and normalizing the column vectors $\{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_n\}$ of W. $\|\cdot\|$ denotes the Euclidean norm (i.e., L_2-norm).

Step 1 (Initialization):

$$\mathbf{u}_1 = \mathbf{w}_1, \ \mathbf{q}_1 = \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|}. \tag{20}$$

Step 2 (Iterative Orthogonalization):

$$\mathbf{u}_{k+1} = \mathbf{w}_{k+1} - \sum_{i=1}^{k}(\mathbf{w}_{k+1}\cdot\mathbf{q}_i)\mathbf{q}_i, \tag{21}$$

$$\mathbf{q}_{k+1} = \frac{\mathbf{u}_{k+1}}{\|\mathbf{u}_{k+1}\|}, \tag{22}$$

where $k = 1, 2, \cdots, n-1$.

Step 3 (The resulting QR factorization in matrix form):

$$\begin{aligned} W &= \left[\mathbf{w_1}, \cdots, \mathbf{w_n}\right] \\ &= \left[\mathbf{q_1}, \cdots, \mathbf{q_n}\right] \begin{bmatrix} \mathbf{w_1}\cdot\mathbf{q_1} & \cdots & \mathbf{w_n}\cdot\mathbf{q_1} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbf{w_n}\cdot\mathbf{q_n} \end{bmatrix} \\ &= QR. \end{aligned} \tag{23}$$

We can express the elements of R are given by:

$$R_{ij} = \begin{cases} \mathbf{w}_i\cdot\mathbf{q}_i = \|\mathbf{u}_i\|, & i = j \\ \mathbf{w}_j\cdot\mathbf{q}_i, & i < j \\ 0, & i > j \end{cases}. \tag{24}$$

From the above derivation process, when $i = j$, $\mathbf{w}_i\cdot\mathbf{q}_i = \|\mathbf{u}_i\|$, this clearly indicates that the diagonal element R_{ii} of the R matrix is the norm of the intermediate orthogonal vector u_i obtained during the Gram-Schmidt orthogonalization process. This norm plays a crucial role in QR decomposition. It reflects the "length" information of the original vector $\mathbf{w}_i$ after progressive orthogonalization processing and has important applications in theoretical analyses based on QR decomposition, such as solving linear equations and computing the eigenvalues of a matrix.

B Proof for diagonal element

The following is a detailed derivation of how $\mathbf{w}_i\cdot\mathbf{q}_i$ equals $\|\mathbf{u}_i\|$ when $i = j$ in the Gram-Schmidt orthogonalization process.

Derivation for the Case of $i = 1$: Given that $\mathbf{u}_1 = \mathbf{w}_1$ and $\mathbf{q}_1 = \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|}$, we calculate $\mathbf{w}_1\cdot\mathbf{q}_1$ according to the definition of the vector inner product:

$$\mathbf{w}_1\cdot\mathbf{q}_1 = \frac{\mathbf{u}_1\cdot\mathbf{u}_1}{\|\mathbf{u}_1\|} = \frac{\|\mathbf{u}_1\|^2}{\|\mathbf{u}_1\|} = \|\mathbf{u}_1\| \tag{25}$$

Derivation for the Case of $i > 1$: Given that $\mathbf{q}_i = \frac{\mathbf{u}_i}{\|\mathbf{u}_i\|}$ and $\mathbf{u}_i = \mathbf{w}_i - \sum_{j=1}^{i-1}(\mathbf{w}_i\cdot\mathbf{q}_j)\mathbf{q}_j$ by the Gram-Schmidt orthogonalization formula, we substitute it into $\mathbf{w}_i\cdot\mathbf{u}_i$:

$$\begin{aligned} \mathbf{w}_i\cdot\mathbf{u}_i &= \mathbf{w}_i\cdot\left(\mathbf{w}_i - \sum_{j=1}^{i-1}(\mathbf{w}_i\cdot\mathbf{q}_j)\mathbf{q}_j\right) \\ &= \mathbf{w}_i\cdot\mathbf{w}_i - \mathbf{w}_i\cdot\left(\sum_{j=1}^{i-1}(\mathbf{w}_i\cdot\mathbf{q}_j)\mathbf{q}_j\right) \\ &= \mathbf{w}_i\cdot\mathbf{w}_i - \sum_{j=1}^{i-1}(\mathbf{w}_i\cdot\mathbf{q}_j)(\mathbf{w}_i\cdot\mathbf{q}_j) \\ &= \mathbf{w}_i\cdot\mathbf{w}_i - \sum_{j=1}^{i-1}(\mathbf{w}_i\cdot\mathbf{q}_j)^2. \end{aligned} \tag{26}$$

Also, since $\mathbf{u}_i = \mathbf{w}_i - \sum_{j=1}^{i-1}(\mathbf{w}_i\cdot\mathbf{q}_j)\mathbf{q}_j$, according to the fact that the square of the vector norm is equal to the inner product of the vector with itself, that is, $\|\mathbf{u}_i\|^2 = \mathbf{u}_i\cdot\mathbf{u}_i$:

$$\begin{aligned} \|\mathbf{u}_i\|^2 &= (\mathbf{w}_i - \sum_{j=1}^{i-1}(\mathbf{w}_i\cdot\mathbf{q}_j)\mathbf{q}_j)\cdot(\mathbf{w}_i - \sum_{k=1}^{i-1}(\mathbf{w}_i\cdot\mathbf{q}_k)\mathbf{q}_k) \\ &= \mathbf{w}_i\cdot\mathbf{w}_i - 2\sum_{j=1}^{i-1}(\mathbf{w}_i\cdot\mathbf{q}_j)(\mathbf{w}_i\cdot\mathbf{q}_j) \\ &\quad + \sum_{j=1}^{i-1}\sum_{k=1}^{i-1}(\mathbf{w}_i\cdot\mathbf{q}_j)(\mathbf{w}_i\cdot\mathbf{q}_k)(\mathbf{q}_j\cdot\mathbf{q}_k). \end{aligned} \tag{27}$$

Since $\{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_n\}$ is a set of orthonormal vectors, when $j \neq k$, $\mathbf{q}_j\cdot\mathbf{q}_k = 0$; when $j = k$, $\mathbf{q}_j\cdot\mathbf{q}_k = 1$. So:

$$\|\mathbf{u}_i\|^2 = \mathbf{w}_i \cdot \mathbf{w}_i - 2\sum_{j=1}^{i-1}(\mathbf{w}_i \cdot \mathbf{q}_j)^2 + \sum_{j=1}^{i-1}(\mathbf{w}_i \cdot \mathbf{q}_j)^2$$
$$= \mathbf{w}_i \cdot \mathbf{w}_i - \sum_{j=1}^{i-1}(\mathbf{w}_i \cdot \mathbf{q}_j)^2. \quad (28)$$

This shows that $\mathbf{w}_i \cdot \mathbf{u}_i = \|\mathbf{u}_i\|^2$. Then

$$\mathbf{w}_i \cdot \mathbf{q}_i = \frac{\mathbf{w}_i \cdot \mathbf{u}_i}{\|\mathbf{u}_i\|} = \frac{\|\mathbf{u}_i\|^2}{\|\mathbf{u}_i\|} = \|\mathbf{u}_i\|.$$

In conclusion, for any i, in the Gram-Schmidt orthogonalization process, when $i = j$, $\mathbf{w}_i \cdot \mathbf{q}_i = \|\mathbf{u}_i\|$.

C Related Work

Other PEFT methods Adapter-based methods incorporated a new trainable module into existing layer in frozen LLMs to reduce training resources but introduced additional inference delay. Typical methods included Adapter (Lester et al., 2021) and Compacter (Mahabadi et al., 2021) that added a linear layer composed of up-sampling, down-sampling, and nonlinear activation function to LLMs, and Compacter shared parameters across layers to further achieve parameter efficiency. Soft prompt-based (Li and Liang, 2021b; Liu et al., 2021; Choi et al., 2023) methods applied soft tokens (prompts) before input or within the attention module to achieve adaptation. However, these prompts were usually sensitive to the parameter initialization and also extended the inference time.

D Details of Benchmarks

D.1 Commonsense Reasoning

The details of the commonsense reasoning benchmarks are as follows:

- **BoolQ**: yes/no questions which are naturally occurring and generated in unprompted and unconstrained settings. The topics of the questions include entertainment media, nature/science, sports, law/government, history, fictional events, and others. There are 3270 questions in the test set.
- **HellaSwag**: commonsense NLI questions including a context and several endings which complete the context. There are 10042 questions in the test set.
- **SIQA**: multiple choice questions for probing emotional and social intelligence in a variety of everyday situations. There are 1954 questions in the test set.
- **PIQA**: questions with two solutions requiring physical commonsense. There are 1838 questions in the test set.
- **ARC-c and ARC-e**: multiple-choice grade-school science questions partitioned into a challenge set and an easy set. The challenge set contains only question answered incorrectly by both a retrieval-based algorithm and a word co-occurrence algorithm. The ARC-c and ARC-e test sets include 1172 and 2376 questions, respectively.
- **OBQA**: multiple-choice questions that require combining a core fact with external common knowledge. There are 500 questions in the test set.
- **WinoGrande**: fill-in-the-blank style questions with two answer choices. There are 1276 questions in the test set.

D.2 Training Corpus for Math, Code, Dialogue, and Cross-task NLU

The details of fine-tuning datasets are as follows:

- **WizardLM-Evol-Instruct**: employs the Evol Instruct strategy to iteratively rewrite seed instructions into increasingly complex variants using LLMs rather than human annotators. It contains 250k instructions in total.
- **MetaMathQA**: a synthetic mathematical reasoning dataset designed to improve LLM performance by bootstrapping questions through question rewriting. It contains 395k problems and corresponding solutions in total.
- **Code-Feedback**: formats interactive code generation sessions in the OpenAI SDK schema. There are 68k multi-turn interactions between users, code models, and compilers.
- **Databricks-dolly-15k**: the first open source, human generated instruction tuning dataset created to enable ChatGPT-style interactivity without proprietary restrictions.

D.3 Validation Benchmarks on Math, Code, Dialogue, and Cross-task NLU

The details of the math, code, dialogue, and cross-task nlu benchmarks are as follows:

Hyperparameter	LoRA	DoRA	HiRA	OLoRA	Pissa	OHoRA
Rank r	16					64
α	32					128
Dropout	0.05					
Batch size	8					
Epochs	3					
Learning rate	2e-4					5e-4
Target module	q, k, v, o, up, down, gate					

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

Hyperparameter	LLaMA-2 7B	Gemma 7B	LLaMA-3.1-8B-I
Batch size	3		
Learning rate	2e-4		
Epoch	1		
Rank r	16		
Rank r_{high}	64	32/48/64	64
α	32/128	32/64/96/128	32/128
Target module	q, k, v, o, up, down, gate		

Table 7: The hyperparameters for various methods for Chat, Math, Code, and Multi-task NLU.

- **MT-Bench**: a specialized evaluation suite of 80 multi turn prompts crafted to test conversational fluency and instruction following capabilities of large language models. The benchmark employs GPT-4 as an automated judge to approximate human preferences, enabling scalable yet alignment sensitive scoring across dialogue turns.

- **GSM8K**: a collection of 8.5k linguistically diverse elementary level math word problems, rigorously authored to require multi step arithmetic reasoning. There are 1k questions in test set.

- **HumanEval**: a hand constructed suite of 164 Python programming challenges, each specified by a function signature, explanatory docstring, and multiple unit tests to validate correct behavior. Problems span tasks in language manipulation, algorithmic logic, and simple mathematics.

- **MMLU (Massive Multitask Language Understanding)**: a comprehensive multiple choice benchmark covering 57 subjects across STEM, humanities, social sciences, and professional fields, totaling 15908 questions.

E Experimental Setup

E.1 Commonsense Reasoning

Table 6 shows the details of hyper-parameters setup for various methods on commonsense reasoning tasks when fine-tuning LLaMA2-7B and LLaMA3-8B.

E.2 Chat, Math, Code, and Multi-task NLU

Following (Meng et al., 2024; Tian et al., 2024), we fine-tune various models on WizardLM-Evol-Instruct, MetaMathQA, Code-Feedback, Dolly-15k datasets and evaluate on MT-Bench, GSM8K, HumanEval, MMLU to test the chat, math, code and multi-task NLU abilities. The detailed implementation is as follows:

- *Chat:* The models are trained on a 100k subset of WizardLM-Evol-Instruct dataset (Xu et al., 2024) and tested on the MT-Bench dataset (Zheng et al., 2023) for conversational abilities. MT-Bench comprises 80 multi-turn questions and GPT-4 is employed to judge the response quality by assigning scores from 1 to 10.

COMPONENT	BOOLQ	PIQA	SIQA	ARC-C	ARC-E	OBQA	HELLAS	WINOG	**AVG ↑**
FC, QKV, GATE, O	**75.96**	**89.99**	82.14	**83.11**	93.14	**88.80**	**96.30**	**89.11**	**87.32**
FC, QKV	72.72	89.50	**82.45**	82.59	**93.43**	88.20	96.09	88.48	86.68
FC	73.58	88.63	80.25	82.25	93.27	85.60	95.66	86.74	85.75
O	72.81	88.90	80.60	80.97	92.17	85.20	95.13	86.66	85.31
QKV	72.45	89.28	80.50	80.89	92.17	85.40	95.34	85.64	85.21
QV	72.23	88.52	80.35	80.46	91.79	85.40	95.25	86.35	85.04
V	69.45	88.36	79.17	80.63	91.62	84.20	94.46	85.00	84.11
QK	70.43	87.16	78.61	78.67	90.70	81.80	93.51	82.08	82.87
Q	67.40	86.24	77.02	79.27	90.24	79.40	92.46	85.00	82.13
GATE	32.66	88.63	79.32	78.84	91.12	83.00	94.17	84.06	78.98
K	62.02	85.15	72.01	76.37	89.10	74.20	90.24	71.82	77.61

Table 8: Performance of OHoRA integrated into different components for fine-tuning LLaMA-3-8B.

COMPONENT	BOOLQ	PIQA	SIQA	ARC-C	ARC-E	OBQA	HELLAS	WINOG	**AVG ↑**
R-KRON	71.28	84.49	80.96	73.81	87.67	81.80	94.06	84.53	82.33
O-KRON	72.26	84.60	80.66	72.78	87.12	82.80	94.23	85.40	82.48

Table 9: Performance comparison between optimized Kronecker product (O-Kron) and original Kronecker product (R-Kron) on commonsense reasoning tasks when fine-tuning LLaMA2-7B.

- *Math:* The models are trained on a 100k subset of MetaMathQA (Yu et al., 2024) and tested on the GSM8K (Cobbe et al., 2021) validation set to assess their mathematical problem-solving abilities. Accuracy is reported as the evaluation metric.

- *Code:* The models are trained on a 100k subset of Code-Feedback (Zheng et al., 2024) and tested on the HumanEval (Chen et al., 2021) for coding ability. HumanEval dataset contains 180 Python programming tasks, and the PASS@1 and PASS@10 are reported as evaluation metrics.

- *Multi-task NLU:* The models are trained on Dolly-15k (Conover et al., 2023) and evaluated with MMLU (Hendrycks et al., 2021) for cross-task high-level NLU. Accuracy is used as the metric.

Table 7 shows the details of hyper-parameters setup for various methods when fine-tuning LLaMA2-7B, Gemma 7B, and LLaMA3.1-8B instruct version.

F Baselines

We select several strong baselines to demonstrate the effectiveness of OHoRA.

LoRA fine-tuned the two low-rank matrices A and B to reparameterize the weight update ΔW. A is initialized using Kaiming initialization and B is initialized with zeros.

OLoRA incorporated orthonormal initialization for the two low-rank matrices A and B, and fine-tuned them to achieve adaptation.

Pissa applied the SVD technique to W_0, and initialized A and B based on the components with larger singular values.

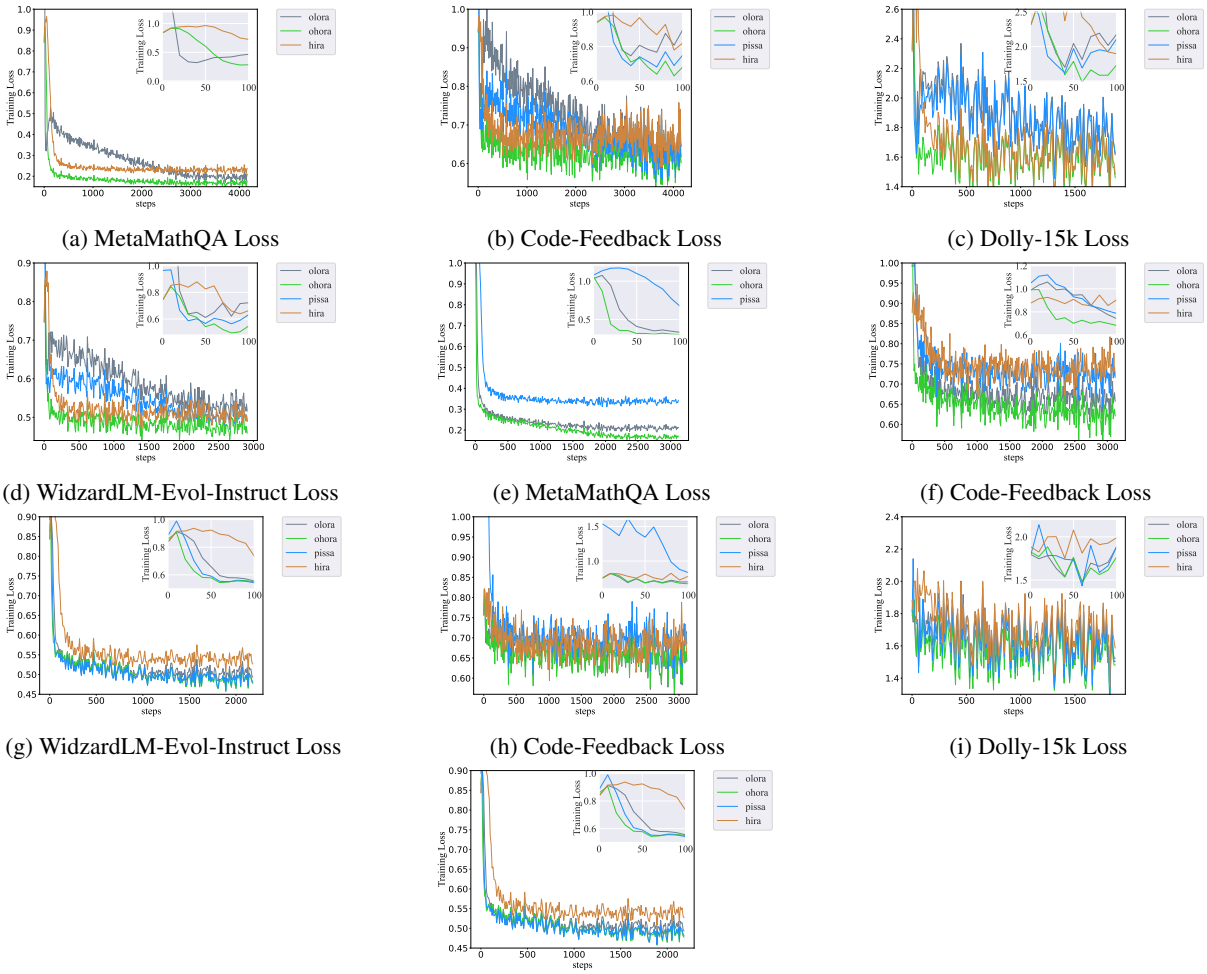
DoRA used weight normalization to decompose the weight matrix into magnitude and direction components. It then employed LoRA for directional updates and treated magnitude as a trained vector to achieve performance resembling FT and ensure parameter efficiency.

HiRA introduced the Hadamard product in combination with W_0 to achieve a high-rank update.

G More Analysis

G.1 Different placement in transformers

As shown in Table 8, applying OHoRA to the FC layers, gate and output projection layers, as well as QKV matrices, achieves the best performance across nearly all tasks compared to alternative component combinations. Notably, fine-tuning only the FC layer or the output projection layer yields performance comparable to that of DORA and Pissa, highlighting the strong representational capacity enabled by OHoRA's dynamic low-redundancy high-rank adaptation.



(a) MetaMathQA Loss (b) Code-Feedback Loss (c) Dolly-15k Loss

(d) WidzardLM-Evol-Instruct Loss (e) MetaMathQA Loss (f) Code-Feedback Loss

(g) WidzardLM-Evol-Instruct Loss (h) Code-Feedback Loss (i) Dolly-15k Loss

(j) WidzardLM-Evol-Instruct Loss

Figure 8: Training loss curves for fine-tuning on three models: panels (a-d) show Gemma-7B on MetaMathQA, Code-Feedback, MMLU and MT tasks; panels (e-g) show LLaMA-2-7B on MetaMathQA, Code-Feedback and MT; panels (h-j) show LLaMA-3.1-8B-I on Code-Feedback, Dolly-15k and MT.

Method	AVG	Time
SVD_{based}	85.50	1375.28s
QR_{based}	85.90	**147.52s**

Table 10: Performance and initialization time cost comparison.

G.2 Effectiveness of QR decomposition

To validate the effectiveness of QR decomposition, QR is replaced with singular value decomposition (SVD) and LLaMA3-8B is fine-tuned on commonsense reasoning tasks. As shown in Table 10, QR decomposition exhibits substantially lower computational time compared to SVD, while achieving marginally higher performance. These results demonstrate the effectiveness of QR decomposition by selecting parameter subspaces that have significant influence on model updates.

G.3 More training loss across different models and tasks

Figure 8 presents additional training loss curves for OHoRA and other baseline methods across different models and tasks, further demonstrating OHoRA's training advantages and its convergence to a better local optimum.

Algorithm 1 OHoRA: An Orthohonal High-Rank Adaptation for Large Language Models

Input: Pre-trained weight matrix $W_0 = [w_1, \ldots, w_n] \in \mathbb{R}^{m \times n}$; Total iterations T; Hyperparameter η.

1. Initialize A_{high} **and** B_{high} **from** W_0**:**

(1) $W_0 \xrightarrow{\text{QR}} Q = [q_1, \ldots, q_n] \in \mathbb{R}^{m \times n}, \quad R \in \mathbb{R}^{n \times n}$.

(2) Let $\lambda = [w_1 \cdot q_1, \ldots, w_n \cdot q_n] = [\lambda_1, \ldots, \lambda_n], r_{high} = \max\{k \in \mathbb{N}^+ \mid k \leq \min\{\sqrt{m}, \sqrt{n}\}, m \bmod k = 0, n \bmod k = 0\}$.

(3) Let π be a permutation of $\{1, 2, \ldots, n\}$ such that $\lambda_{\pi(1)} \geq \lambda_{\pi(2)} \geq \cdots \geq \lambda_{\pi(n)}$.

(4) Define index sets: $I_1 = \{\pi(i) \mid 1 \leq i \leq r_{high}\}, I_2 = \{\pi(i) \mid r_{high} < i \leq 2\, r_{high}\}$.

(5) Let $Q_1 = Q_{:,I_1}, R_1 = R_{I_1,:}, Q_2 = Q_{:,I_2}, R_2 = R_{I_2,:}$. And A_{high} and B_{high} are defined as:

$$A_{high} = Q_{1[:,m/r_{high}]} R_{1[:,:r_{high}]}, B_{high} = Q_{2[:n/r_{high}]} R_{2[:,:r_{high}]}.$$

(6) Update W_0 to W_{res}: $W_{res} = W_0 - A_{high} \otimes B_{high}^T$.

2. for $t = 1, \ldots, T$ **do**

(1) Reshape input x: $x \in \mathbb{R}^n \to Z \in \mathbb{R}^{\frac{n}{r_{high}} \times r_{high}}$.

(2) Forward pass: $y = W_{res}x + \text{vec}\big(B_{high}^\top Z A_{high}^\top\big)$.

(3) Update A_{high} and B_{high}:

$$A_{high}^{(t)} = A_{high} - \eta \textstyle\sum_{i=1}^{t-1} (y - y_{true}) \big(I_n \otimes (B_{\text{high}}^{(i)})^\top Z\big),$$
$$B_{high}^{(t)} = B_{high} - \eta \textstyle\sum_{i=1}^{t-1} (y - y_{true}) \big(A_{\text{high}}^{(i)} Z^\top \otimes I_n\big).$$

End for

3. Output: The fine-tuned parameters $\{A_{high}^{(T)}, B_{high}^T\}$.