R-LoRA: Randomized Multi-Head LoRA for Efficient Multi-Task Learning

Anonymous ACL submission

Abstract

Fine-tuning large language models (LLMs) is computationally expensive, and Low-Rank Adaptation (LoRA) provides a cost-effective solution by approximating weight updates through low-rank matrices. In real-world scenarios, LLMs are fine-tuned on data from multiple domains to perform tasks across various fields, embodying multi-task learning (MTL). LoRA often underperforms in such complex scenarios. To enhance LoRA's capability in multi-task learning, we propose R-LoRA, which incorporates Multi-Head Randomization. Multi-Head Randomization diversifies the head matrices through Multi-Head Dropout and Multi-Head Random Initialization, enabling more efficient learning of task-specific features while maintaining shared knowledge representation. Our approach not only improves performance in MTL but also reduces GPU memory usage and training time. Experiments show that R-LoRA's gains stem from increased diversity in the head matrices, demonstrating its effectiveness for multi-task learning. The code will be open-sourced after the revision.

1 Introduction

In recent years, large language models (LLMs) have manifested unprecedentedly superior performance in various natural language processing (NLP) tasks (Brown, 2020; Zhao et al., 2023; Chang et al., 2024). Due to its impressive capabilities in language understanding and generation, LLMs have gained extensive interest from both academia and industry. Despite their high generalizability, LLMs still require fine-tuning for specific domains or updating the knowledge base (Agiza et al., 2024; Xin et al., 2024).

Supervised fine-tuning (SFT) is crucial for aligning large language models (LLMs) with human instructions, which trains the model with a small yet high-quality set of labeled data (Hu et al., 2021; Xia et al., 2024). The vast number of parameters in LLMs poses significant challenges regarding computational efficiency and memory consumption during full fine-tuning (FT), which updates all parameters.

To address the issue of hardware requirements for LLM adaptation, a solution called parameter efficient fine-tuning (PEFT) has been proposed (Han et al., 2024). PEFT methods reduce VRAM usage of cached optimizer states by only optimizing a fraction of model parameters while keeping the rest frozen. Various PEFT methods have been widely studied(Li and Liang, 2021)(Liu et al., 2024c)(Liu et al., 2022)(Hu et al., 2021). Among these methods, LoRA has emerged as the mainstream alternative to full parameter fine-tuning. Instead of updating the original parameter matrix directly, LoRA approximates the updated parameters using the product of two smaller matrices. During inference, the output obtained from the original parameter matrix is combined with the output from the updated parameter matrices. However, LoRA does not perform well in multi-task scenarios, particularly in dealing with complex datasets.

Recent LoRA variants have improved multi-task learning by employing multiple LoRA adapters, including Multi-LoRA (Wang et al., 2023), LoRA-MoE (Dou et al., 2023), and MoeLoRA (Liu et al., 2024a). We refer to this extended framework as the Multi-Adapter LoRA architecture, which consists of multiple down-projection matrices (A) and their corresponding head matrices (B), enabling task-specific adaptation through diverse parameter sets. Notably, LoRA-MoE and MoeLoRA further enhance this architecture by introducing a Mixture of Experts (MoE) mechanism to aggregate adapter outputs. Tian et al. (2024) observes that in the Multi-Adapter LoRA architecture, the parameters of the down-projection matrices A are relatively consistent, while the differences between the head matrices B are more pronounced, which

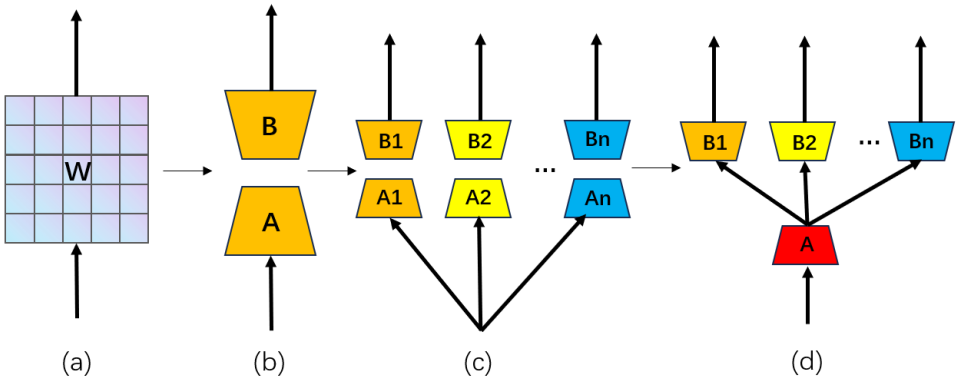

Figure 1: Training architecture comparison. (a) Full parameter fine-tuning; (b) Vanilla LoRA; (c) Multi-Adapter architecture; (d) Multi-Head/Asymmetric architecture.

aids in capturing task-specific knowledge. To leverage this property, HydraLoRA (Tian et al., 2024) is proposed to feature an asymmetric architecture with one shared down-projection matrix A and multiple task-specific head matrices B. Additionally, HydraLoRA also employs an MoE mechanism to aggregate the outputs of the head matrices. This design achieves a good balance between training performance and parameter efficiency. In this work, we explicitly define asymmetric architecture as a Multi-Head structure, and introduce Multi-Head randomization to improve LLMs' performance on multi-task learning. The mathematical formalization of the Multi-Head structure is detailed in Section 2.2. Figure 1 illustrates the differences among the aforementioned structures.

However, in the Multi-Head architecture, the parameter similarity among head matrices remains high, hindering task-specific knowledge learning. This is due to the zero initialization of head matrices B, leading to similar update directions. To address this limitation, R-LoRA employs multi-head randomization, combining random initialization with multi-head dropout. This approach diversifies both the starting points and inputs of the head matrices, enabling more effective task-specific learning by breaking initial symmetry and promoting distinct optimization trajectories. Our work makes the following key contributions:

- *We reveal redundancy and symmetry in the head matrices of Multi-Head LoRA, limiting its ability to capture diverse task-specific knowledge.*

- *We propose R-LoRA, introducing Multi-Head Randomization to enhance both performance and efficiency in multi-task learning.*

- *Extensive experiments validate R-LoRA's superiority, with analysis showing performance gains stem from diversified head matrices.*

2 Related Works

2.1 LoRA

Current LLMs generally follow a decoder-only structure, characterized by a series of blocks, each comprising two key components with residual connections: a multi-head self-attention (MHA) layer and a feed-forward network (FFN) (Vaswani, 2017). These layers involve using dense learnable matrices.

There is a need to adapt LLMs for specific tasks or domains with limited resources. To achieve this, low-rank adaptation (LoRA) (Hu et al., 2021), inspired by the concept of low intrinsic dimensionality in LLMs, decomposes the weight gradient $\Delta\mathbf{W}$ into low-rank matrices, thereby reducing the number of trainable parameters. Specifically, for a dense weight matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$, LoRA employs two low-rank matrices, $\mathbf{B} \in \mathbb{R}^{m\times r}$ and $\mathbf{A} \in \mathbb{R}^{r\times n}$, to approximate the accumulated gradient updates $\Delta\mathbf{W}$. The rank r is chosen to be much smaller than the minimum of d and k, effectively decreasing the number of trainable parameters from $m \times n$ to $2r(m \times n)$. Consequently, the resulting weight matrix is expressed as $\mathbf{W} + \mathbf{BA}$, and the output h for an input x through this updated weight matrix is formulated as:

$$h = (\mathbf{W} + \Delta\mathbf{W})x = \mathbf{W}x + \mathbf{BA}x \quad (1)$$

Typically, matrix B is initialized with zeros, while matrix A is initialized using Kaiming Uniform (He et al., 2015). This initialization strategy ensures that the initial outputs remain consistent with the pre-trained model, thereby avoiding the introduction of random disturbances.

Following LoRA, AdaLoRA (Zhang et al., 2023) dynamically learns the rank size needed for LoRA in each layer of the model. DeltaLoRA (Zi et al., 2023) updates the original weights of the model using parameters from adapter layers, enhancing LoRA's representational capacity. DoRA (Liu et al., 2024b) introduces a magnitude component to learn the scale of ΔW while utilizing the original AB as a direction component of ΔW. PiSSA (Meng et al., 2025) and LoRA-GA (Wang et al., 2024) have improved the convergence speed and performance of LoRA by refining its initialization method. Their approaches focus on optimizing the initial parameter settings, which enhances the training dynamics and leads to more efficient and stable convergence.

2.2 Multi-Head architecture

MTL-LoRA (Yang et al., 2024) and HydraLoRA (Tian et al., 2024) are pioneering methods that introduce the multi-head architecture into LoRA. This architecture is characterized by a central shared down-projection matrix A and multiple distinct head matrices B, enabling efficient and flexible adaptation across diverse tasks. As shown in Figure 1, this architecture differentiates task-specific information while effectively capturing shared knowledge across various tasks. The Multi-Head architecture can be formulated as:

$$W + \Delta W = W + \sum_{i=1}^{N} \omega_i \cdot B_i A \tag{2}$$

In MTL-LoRA (Yang et al., 2024) and HydraLoRA (Tian et al., 2024), the weights w_i are computed through the routing matrix W_r and the softmax function. It can be formulated as:

$$\omega = Softmax(W_r x) \tag{3}$$

2.3 Dropout

Dropout is a widely used technique to prevent overfitting in deep networks by randomly deactivating units during training (Srivastava et al., 2014). This process samples from an exponential number of thinned networks, reducing unit co-adaptation and enhancing noise robustness. The following is the formulation of Dropout.

1. **Mask vector:** Generate a binary mask vector $\mathbf{m} \in \{0,1\}^d$, where each element m_j independently takes the value 1 with probability $1-p$ and 0 with probability p: $m_j \sim \text{Bernoulli}(p), \quad j = 1, \ldots, d$

2. **Apply the mask:** During training, multiply the input $\mathbf{x}$ or the activation values by the mask $\mathbf{m}$ element-wise to get the masked output $\tilde{\mathbf{x}}$: $\tilde{\mathbf{x}} = \mathbf{m} \odot \mathbf{x}$, where $\odot$ denotes the Hadamard product (element-wise multiplication).

3. **Scale activation values:** To maintain consistent expected outputs between training and testing, the retained neurons are typically scaled (multiplied by $\frac{1}{1-p}$): $\tilde{\mathbf{x}} = \frac{1}{1-p}\mathbf{m} \odot \mathbf{x}$

Dropout operations require both the computation and storage of masking vectors during training. At test time, the full network is utilized, benefiting from the ensemble effect of the thinned networks. In our work, we adapt dropout to a novel context within the multi-head structure of R-LoRA. Specifically, we employ dropout to differentiate the inputs of the head matrices, ensuring that each head learns distinct and complementary representations while also reducing computational overhead.

3 Motivation

In this section, we analyze the parameter similarity between different head matrices in the Multi-Head LoRA architecture. To achieve our objectives, we focus on HydraLoRA (Tian et al., 2024) and use cosine similarity and the T-SNE method to observe the parameters of the head matrices. We fine-tune Qwen2.5-3B (Qwen Team, 2024) with HydraLoRA (Tian et al., 2024) on five different tasks. The details of the dataset can be referred to Appendix B.1. First, we flatten the head matrices into vectors and then calculate the cosine similarity between the vectors to obtain a similarity matrix. The average value of the matrix is regarded as the similarity of the head matrix corresponding to the parameter matrix. Additionally, we perform T-SNE analysis on all the head matrices in Figure 6 of Appendix A.

As shown in Figure 2, the average similarity between different head matrices still reaches around 80%. With such a high similarity, the knowledge learned between different head matrices is also quite similar, which hinders the learning of task-specific knowledge. To the best of our knowledge, this is due to the zero initialization of the head matrices. Tuning a pretrained LLM essentially becomes optimizing in a much smaller parameter space around the local optimum of pretrained models. After receiving the outputs from the shared down-projection matrix A, the outputs of the head

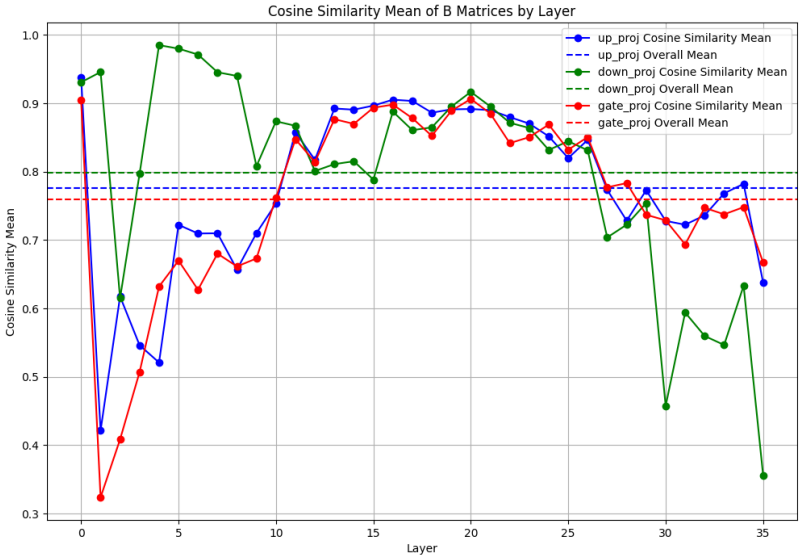

Figure 2: Cosine similarity among head matrices. "Overall mean" represents the average similarity across all layers.

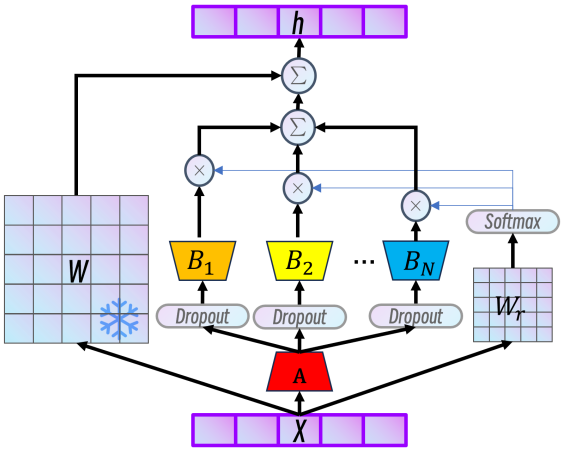

Figure 3: Overview of the R-LoRA.

matrices are highly similar in the early stages of training, leading to highly similar update directions during gradient updates.

Research Question 1: *Is there a simple yet effective approach to differentiate head matrices such that they capture distinct task-specific knowledge for efficient multi-task learning?*

4 Method

In this work, we propose R-LoRA, which leverages multi-head randomization to assist the model in learning distinct knowledge. Multi-head randomization consists of two components: multi-head dropout and random initialization. An overview of R-LoRA is illustrated in Figure 3

Research Objective: *To exploit randomization to differentiate the head matrices, thereby facilitating the optimization of their parameters to distinct regions and enhancing the diversity among the head matrices.*

4.1 Multi-Head Dropout

Multi-Head LoRA architecture is characterized by a shared down-projection matrix A and several distinct head matrices B. In HydraLoRA (Tian et al., 2024), the head matrices receive the same output from the shared matrix A. According to (Hayou et al., 2024) and (Tian et al., 2024), the down-projection matrix A and the head matrix B in LoRA play distinct roles. Specifically, the down-projection matrix A primarily captures task-agnostic knowledge, encoding generalizable features applicable across tasks, while the head matrices specialize in task-specific knowledge, enabling the model to adapt to the unique requirements of individual tasks. This division of roles enhances the model's ability to balance generalization and specialization in multi-task learning scenarios. We propose employing multi-head dropout to differentiate the outputs of the down-projection matrix A, thereby ensuring that the head matrices produce distinct outputs. The framework of Multi-Head dropout and R-LoRA is shown in Figure 3. Our architecture builds upon the multi-head structure of LoRA by incorporating multi-head dropout. After the input is processed by the down-projection ma-

Method	A Init	B Init
LoRA	$U\left(-\sqrt{\frac{3}{d_{in}}}, \sqrt{\frac{3}{d_{in}}}\right)$	0
HydraLoRA	$U\left(-\frac{1}{d_{in}}, \frac{1}{d_{in}}\right)$	0
R-LoRA	$\frac{\sqrt[4]{d_{out}}}{\sqrt{\gamma}} \cdot N\left(0, \frac{1}{d_{in}}\right)$	$\frac{\sqrt[4]{d_{out}}}{\sqrt{\gamma}} \cdot N\left(0, \frac{1}{d_{out}}\right)$

Table 1: Comparison of initialization.

trix A, it generates a task-agnostic representation. Multi-head dropout then diversifies this representation, allowing the model to learn task-specific knowledge from multiple perspectives and improving both generalization and task adaptability.

Additionally, R-LoRA's multi-head dropout mechanism offers practical advantages by reducing computational overhead and memory usage. Unlike the original LoRA and Multi-Head structure LoRA, which perform dropout on the input $X \in \mathbb{R}^{b\times m}$, R-LoRA applies Multi-Head Dropout to the intermediate representations. $H \in \mathbb{R}^{b\times r}, r \ll m$. Since dropout operations require both the computation and storage of masking matrices, applying dropout to the lower-dimensional H results in reduced computational costs and lower GPU memory consumption.

4.2 Multi-Head Random Initialization

The zero initialization of the head matrices results in identical starting points for the different head matrices during training, causing them to converge to similar positions. As demonstrated in Table 1, To address this limitation, we adopt random initialization to break the symmetry of initial head matrices and diversify optimization trajectories, thereby encouraging them to converge to different positions. To stabilize the magnitude of outputs and enhance model performance, we incorporate a scaling coefficient into the initialization process of the head matrices. Specifically, inspired by (He et al., 2015) and (Wang et al., 2024), we introduce a coefficient $\frac{\sqrt[4]{d_{out}}}{\sqrt{\gamma}}$ or $\frac{\sqrt[4]{d_{in}}}{\sqrt{\gamma}}$ during initialization to the matrices to ensure scale stability. The γ is a hyperparameter set to 64 based on empirical findings from (Wang et al., 2024). Notably, (Wang et al., 2024) theoretically analyzes that such a scaling factor helps maintain the numerical stability of the output magnitudes throughout training. When the head matrices are initialized with non-zero values, the initial $\Delta\mathbf{W_0}$ is no longer zero. To maintain consistency with the pre-trained model's initial outputs and avoid random perturbations, we subtract it from the original parameter matrix $\mathbf{W}$ during the initialization phase. It can be formulated as:

$$W = W - \Delta\mathbf{W_0} = W - \frac{1}{N}\sum_{i=1}^{N} B_{i0} \cdot A_0 \quad (4)$$

5 Experiments

In this section, we validate the effectiveness of R-LoRA across various models and settings. Specifically, we evaluate R-LoRA's multi-task adaptability using Qwen2.5 (Qwen Team, 2024) in Setting 1 and its multi-task generalization capability using LLaMA-2 (Touvron et al., 2023) in Setting 2. Model sizes range from 3B to 13B. An extensive ablation study further demonstrates the effectiveness of multi-head randomization in R-LoRA.

5.1 Experiment Settings

Datasets & Benchmarks: Setting 1: We fine-tune Qwen2.5 on datasets covering commonsense and mathematical reasoning tasks and evaluate performance on their respective test sets. **Setting 2:** We fine-tune LLaMA-2 on a subset of the Flanv2 dataset (Brown, 2020), which includes tasks grouped into 10 distinct task clusters. Performance is measured using the Big-Bench Hard (BBH) benchmark. Additional details about the datasets and implementation details can be found in Appendix B and Appendix C, respectively.
Baselines: In Setting 1, we compare R-LoRA with LoRA, Multi-LoRA, MoeLoRA, and HydraLoRA. In Setting 2, the comparison includes LoRAHub (Huang et al., 2023), which employs black-box optimization for weighted averaging of LoRAs, LoRA-MoE (Liu et al., 2024a), which integrates lightweight experts with a Mixture of Experts architecture, and HydraLoRA.

5.2 Performance of R-LoRA on Multi-Tasks

The evaluation across diverse multi-task reasoning datasets, as shown in Table 2 and Table 3,

Schemes	Task1	2	3	4	5	6	7	8	Avg	%Par	A	B
Qwen2.5-3B												
LoRA*1	80.00	56.50	84.80	72.10	90.10	87.60	87.60	44.15	75.36	0.18	1	1
LoRA*2	86.30	56.40	84.70	72.60	91.40	87.90	87.60	44.80	76.46	**0.45**	1	1
Multi-LoRA	84.50	55.40	82.70	72.10	89.80	81.80	87.69	44.80	74.85	0.60	3	3
MoeLoRA	87.40	58.10	85.60	73.40	92.25	87.40	87.34	45.50	77.12	0.68	3	3
HydraLoRA	86.50	56.40	85.00	73.40	92.00	87.40	88.38	45.10	76.77	0.45	1	3
R-LoRA	87.10	57.90	88.13	73.90	94.70	88.25	88.26	45.60	**77.98**	**0.45**	1	3
Qwen2.5-7B												
LoRA*1	87.20	59.85	87.60	80.10	91.10	89.50	90.30	47.80	79.18	0.10	1	1
LoRA*2	88.40	60.80	88.40	81.50	93.60	91.20	91.80	48.10	80.48	**0.25**	1	1
Multi-LoRA	88.30	58.90	87.50	79.80	91.50	88.40	91.90	47.90	79.28	0.33	3	3
MoeLoRA	89.50	61.40	88.90	82.90	93.60	91.50	91.90	48.70	81.05	0.38	3	3
HydraLoRA	88.60	61.20	89.50	81.70	93.60	91.60	91.70	48.10	80.75	0.25	1	3
R-LoRA	89.80	62.50	89.40	83.70	95.10	92.10	92.17	50.80	**81.95**	**0.25**	1	3

Table 2: Comparison of different training schemes on multi-task reasoning datasets. The rank of LoRA*2 was set to 10 to ensure that its trainable parameters matched those of R-LoRA, while all other configurations used a rank of 4.

Metrics	Base	LoRA	LoRAHub*	LoRA MoE*	HydraLoRA	R-LoRA
7B	31.6	37.1	39.7	40.3	41.5	**42.2**
13B	38.4	40.8	41.9	43.7	44.2	**45.1**
A/B for training	-	1/1	48/48	48/48	1/10	1/10
A/B for inference	-	1/1	20/20	48/48	1/10	1/10
% Param	-	0.062	1.240	2.976	0.341	0.341

Table 3: Comparison of different training schemes on multi-tasks. * indicates results from (Tian et al., 2024).

demonstrates that R-LoRA achieves superior performance compared to all other methods. By introducing multi-head randomization, R-LoRA achieves significantly improved multi-task adaptability and generalization capabilities. The performance gains achieved by R-LoRA, driven by these innovations, outperform LoRA and SOTA multi-head LoRA methods like HydraLoRA. This highlights R-LoRA's enhanced generalization and task adaptability. Additional results on the performance of R-LoRA under single-task settings are provided in Appendix A.

5.3 Efficiency of R-LoRA

We conducted a comparative analysis of the memory usage and training time between R-LoRA and the original Multi-head structure LoRA using Qwen2.5-3B under various configurations. Table 4 demonstrates that R-LoRA's multi-head dropout approach reduces GPU memory consumption by up to 20% and cuts training time by up to 8%, highlighting its superior efficiency in comparison to traditional methods.

5.4 Parameter Analysis

Research Question2: *Does multi-head randomization effectively enhance the acquisition of diverse knowledge across the head matrices?*

In this section, we analyze the parameter differences among the head matrices in R-LoRA. The methodology and experimental setup align with those described in Section 3. As shown in Figure 4, the parameter similarity between head matrices in R-LoRA is reduced to below 70%. This significant decrease indicates that multi-head randomization effectively enhances the model's capacity to learn task-specific knowledge, thereby mitigating redundant learning and increasing the diversity of acquired knowledge across tasks.

5.5 Training Process

Research Question3: *Does multi-head randomization impact the stability of the training process?*

As illustrated in Figure 5, R-LoRA benefits from multi-head randomization, exhibiting significantly

Schemes	3 Heads(bfloat16)	5 Heads(bfloat16)	3 Heads(float32)	5 Heads(float32)
MD	18.53GB / 2.20h	22.05GB / 2.75h	34.23GB / 8.68h	41.24GB / 9.65h
ID	23.42GB / 2.41h	30.25GB / 3.25h	42.09GB / 9.08h	54.45GB / 10.31h

Table 4: Comparison of memory consumption and per-epoch training time across different dropout operations. MD denotes our proposed Multi-Head Dropout, while ID represents input dropout applied to x in HydraLoRA.

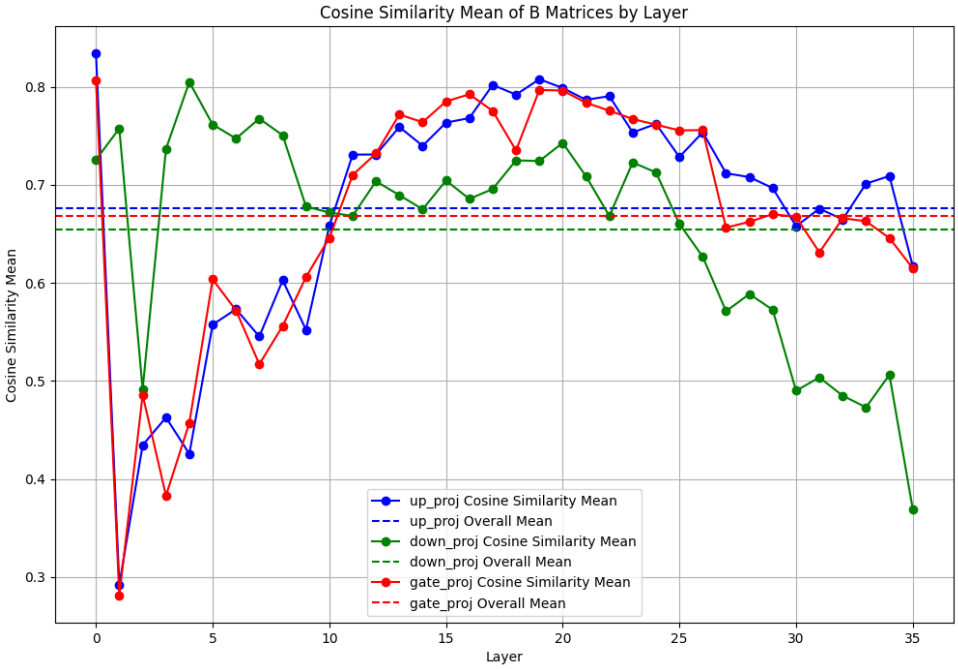

Figure 4: Cosine similarity among head matrices in R-LoRA. "Overall mean" represents the average similarity across all layers.

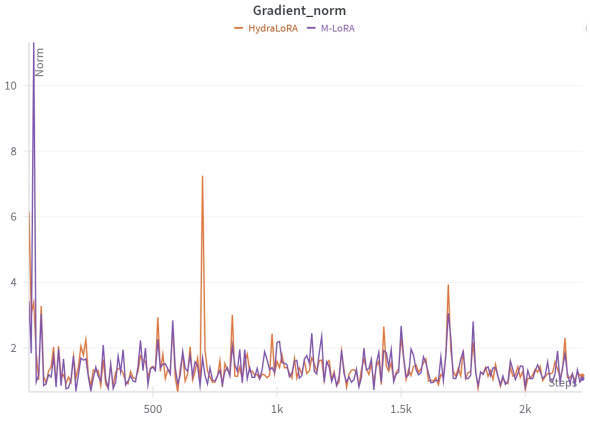

Figure 5: Gradient norm dynamics during training: Comparison of conventional and multi-head randomized configurations, highlighting enhanced stability through diversified head matrices.

larger gradient norms in the early stages of training compared to HydraLoRA. This drives the head matrices to converge to distinct regions, enhancing the model's ability to capture diverse representations and improving overall performance. Furthermore, R-LoRA exhibits superior training stability, as evidenced by its more stable gradient norms throughout the training process. This stability enables the model to effectively acquire diverse knowledge without compromising training efficiency and robustness.

5.6 Ablation Study

Ablation studies were conducted on Llama3.2-3B and Qwen2.5-3B models across eight-task configurations, using 11 datasets spanning 8 categories. All models were evaluated on their respective test sets, with results summarized in Table 5. Dataset details are provided in the Appendix B.4. More results on the smaller model Qwen2.5-0.5B are shown in the Appendix A

Experimental results demonstrate that the two key components of multi-head randomization—random initialization and dropout—are pivotal for enhancing the model's adaptability across tasks. Multi-head randomization remains effective even when initialization is isolated to LoRA B. As shown in Table 5, R-LoRA with zero-

Schemes	Task1	2	3	4	5	6	7	8	Avg
Llama3.2-3B									
R-LoRA	95.82	83.68	84.25	85.48	71.45	74.12	84.34	85.39	**83.07**
w/o MD	95.24	83.25	82.46	84.75	70.27	73.96	84.57	83.29	82.22
w/o MI	94.66	82.77	83.58	83.25	69.79	74.23	83.28	84.16	81.97
Zero A	95.67	83.46	83.47	85.64	70.27	73.84	84.13	84.89	<u>82.67</u>
HydraLoRA	95.12	82.14	83.88	82.68	69.86	72.33	79.25	84.25	81.19
Qwen2.5-3B									
R-LoRA	96.42	83.27	85.34	86.49	72.84	73.86	86.24	88.94	**84.18**
w/o MD	96.22	83.66	83.25	84.72	71.15	73.24	85.02	88.12	83.17
w/o MI	96.10	83.21	83.65	84.50	71.69	72.05	83.24	88.36	82.85
Zero A	96.24	84.02	84.36	85.89	72.13	73.51	85.26	89.13	<u>83.82</u>
HydraLoRA	95.89	83.53	82.97	84.24	70.95	71.93	83.06	87.33	82.49

Table 5: Results of Ablation Studies on Qwen2.5 and Llama3.2 with Different Schemes Across Various Tasks. The table compares R-LoRA with its ablated versions (without Multi-Head Dropout/MD, without Multi-Head Initialization/MI, and zero initialization to LoRA A in R-LoRA) against HydraLoRA across eight tasks.

initialized LoRA A (Zero A) consistently outperforms HydraLoRA, demonstrating that the performance gains are primarily attributed to the diversified parameter spaces in the head matrices B. Random initialization assigns unique weights to each head matrix, enabling task-specific pattern capture. Dropout diversifies inputs to the head matrices, promoting distinct learning pathways. Together, these components improve task-specific feature capture while maintaining robustness in multi-task learning.

6 Discussion

In this section, we discuss the underlying mechanisms behind R-LoRA's multi-head randomization that enhance multi-task learning. As revealed in the Motivation section, a key limitation of traditional multi-head LoRA lies in the high similarity among head matrices, which stems from zero-initialization. This initialization scheme confines heads to symmetric states, limiting their ability to explore sparse yet critical subspaces, such as those relevant to rare syntactic relationships. Consequently, heads tend to converge on overlapping subspaces, failing to adequately address task-specific requirements.

Each token's semantic logics (e.g., semantic, syntactic, contextual) naturally reside in multiple subspaces of high-dimensional embedding spaces. The Multi-Head mechanism excels by decomposing these logics into distinct subspaces for independent processing. R-LoRA aligns with this principle through two key innovations:

- **Multi-Head Dropout**: Promotes heterogeneous feature learning to capture complementary aspects of the embedding space.

- **Multi-Head Random Initialization**: Decouples head trajectories to prevent convergence to overlapping subspaces.

As shown in Figure 4, R-LoRA effectively reduces the similarity among head matrices, promoting diverse feature learning across tasks. Empirical results in Table 5 further confirm that the performance gains are primarily attributed to the diversified parameter spaces in LoRA B, rather than LoRA A, highlighting the importance of the up-projection module in enabling task-specific adaptation.

7 Conclusion

In this work, we first analyze the multi-head structure of LoRA, revealing excessive similarity among head matrices that limits task-specific learning. To address this, R-LoRA introduces multi-head randomization, a simple yet effective approach that differentiates head matrices, enabling the model to learn diverse knowledge across tasks. This innovation enhances both performance and efficiency, reducing GPU memory usage and training time.

Extensive experiments validate R-LoRA's superiority. The performance gains stem primarily from increased diversity in the parameter spaces of the head matrices, confirming the effectiveness of R-LoRA for efficient multi-task learning.

8 Limitation

Despite the promising results of R-LoRA, several limitations should be acknowledged. While we have conducted extensive experiments to validate its effectiveness, the inherent complexity of multi-task learning highlights the importance of further exploration and broader evaluation. Currently, our validation focuses on NLP tasks, and extending the method to other modalities, such as computer vision and multimodal settings, represents an exciting avenue for future research. These directions could help unlock the full potential of R-LoRA and deepen our understanding of its applicability across diverse domains.

References

Ahmed Agiza, Marina Neseem, and Sherief Reda. 2024. Mtlora: Low-rank adaptation approach for efficient multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16196–16205.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45.

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

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, et al. 2023. Loramoe: Revolutionizing mixture of experts for maintaining world knowledge in language model alignment. *arXiv preprint arXiv:2312.09979*, 4(7).

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

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

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

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2023. Lorahub: Efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv:2307.13269*.

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*.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

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. *Preprint*, arXiv:2310.18339.

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. *arXiv preprint arXiv:2402.09353*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2024c. Gpt understands, too. *AI Open*, 5:208–215.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2025. Pissa: Principal singular values and singular vectors adaptation of large language models. *Advances in Neural Information Processing Systems*, 37:121038–121072.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Chengzhong Xu. 2024. Hydralora: An asymmetric lora architecture for efficient fine-tuning. *Preprint*, arXiv:2404.19245.

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

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Alex Wang. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Shaowen Wang, Linxi Yu, and Jian Li. 2024. Lora-ga: Low-rank adaptation with gradient approximation. *Preprint*, arXiv:2407.05000.

Yiming Wang, Yu Lin, Xiaodong Zeng, and Guannan Zhang. 2023. Multilora: Democratizing lora for better multi-task learning. *arXiv preprint arXiv:2311.11501*.

Tingyu Xia, Bowen Yu, Kai Dang, An Yang, Yuan Wu, Yuan Tian, Yi Chang, and Junyang Lin. 2024. Rethinking data selection at scale: Random selection is almost all you need. *arXiv preprint arXiv:2410.09335*.

Chunlei Xin, Yaojie Lu, Hongyu Lin, Shuheng Zhou, Huijia Zhu, Weiqiang Wang, Zhongyi Liu, Xianpei Han, and Le Sun. 2024. Beyond full fine-tuning: Harnessing the power of lora for multi-task instruction tuning. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 2307–2317.

Yaming Yang, Dilxat Muhtar, Yelong Shen, Yuefeng Zhan, Jianfeng Liu, Yujing Wang, Hao Sun, Denvy Deng, Feng Sun, Qi Zhang, Weizhu Chen, and Yunhai Tong. 2024. Mtl-lora: Low-rank adaptation for multi-task learning. *Preprint*, arXiv:2410.09437.

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

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

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

A More Results

A.1 T-SNE analysis

The T-SNE analysis of head matrices in HydraLoRA is shown in Figure 6.

A.2 Performance of R-LoRA on Single Task

We compare R-LoRA against various PEFT methods on single datasets: 1) Full fine-tuning; 2) Prompt Tuning (Lester et al., 2021); 3) P-Tuning (Liu et al., 2024c); 4) Prefix Tuning (Li and Liang, 2021); 5) IA^3 (Liu et al., 2022); 6) AdaLoRA (Zhang et al., 2023); 7) HydraLoRA (Tian et al., 2024).

As shown in Table 6, in the single-task setting, where the knowledge and text format of the data are relatively homogeneous, R-LoRA demonstrates slightly improved performance compared to HydraLoRA. While multi-head randomization is primarily designed for multi-task learning, its ability to learn diverse knowledge remains beneficial even in single-task scenarios. This slight edge over HydraLoRA underscores R-LoRA's capacity to capture varied patterns effectively, even when its full potential is not fully utilized in single-dataset settings. These results further highlight R-LoRA's robustness and adaptability across different task complexities.

A.3 Ablation study of R-LoRA on smaller model

Table 7 show the ablation study on Qwen2.5-0.5B

A.4 Datasets in Single-task

1. **General**: We fine-tune with the general instruction tuning dataset `databricks-dolly-15k` for generic language capability and evaluate with MMLU.

2. **Medical**: We fine-tune with `GenMedGPT` and `clinic-10k` from `ChatDoctor` for medicine applications and evaluate medical tasks in MMLU including three related tasks: "clinical knowledge", "professional medicine", and "college medicine".

3. **Law**: We fine-tune with two legal instruction tuning datasets `Lawyer-Instruct` and `US-Terms` then evaluate with law tasks in MMLU including two related tasks: "professional law" and "international law".

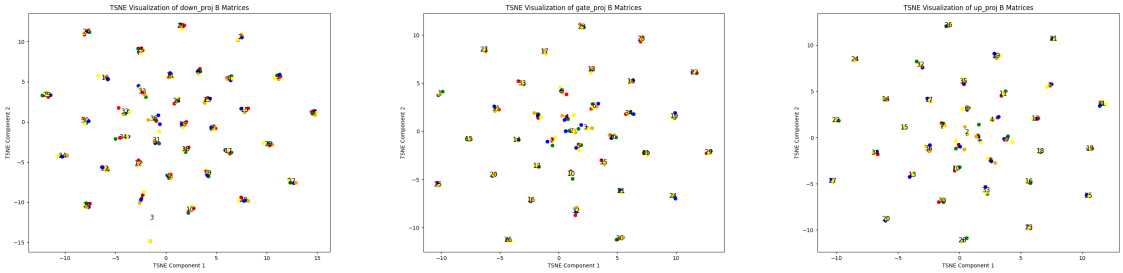

Figure 6: T-SNE analysis of head matrices in HydraLoRA

4. **Math**: We fine-tune with the training split of `GSM8K` for mathematical reasoning and evaluate with the test set of `GSM8K`.

5. **Code**: We fine-tune with `CodeAlpaca` for code generation and evaluate with `HumanEval`.

B Datasets

B.1 Motivation

In the section of Motivation, We fine-tune Qwen2.5-3B on five tasks: Paraphrase Detection (QQP), Natural Language Inference (QNLI) (Wang, 2018), Commonsense Reasoning (SIQA) (Sap et al., 2019), Physical Commonsense Reasoning (PIQA) (Bisk et al., 2020), and Math (GSM8K) (Cobbe et al., 2021)

B.2 Setting 1

1. **Reading Comprehension**: BoolQ
2. **Science Question Answering**: SiQA
3. **Physical Question Answering**: PiQA
4. **Word Relation Reasoning**: Winogrande
5. **Commonsense Reasoning**: Hellaswag
6. **Open-Book Question Answering**: OBQA
7. **Closed-Book Question Answering**: ARC
8. **Mathematical Reasoning**: GSM8K

B.3 Setting 2

Following (Tian et al., 2024), for complex mixed multi-task/domain, we select a portion of the `Flanv2` datasets covering Natural Language Understanding (NLU) and Natural Language Generation (NLG), which can be grouped into 10 distinct task clusters. Then we evaluate it with the Big-Bench Hard (BBH) benchmark.

We summarize the details of the used datasets as follows:

1. **Struct-to-Text Conversion**: This task evaluates the capability to generate natural language descriptions from structured data inputs. We use the following datasets: (1) CommonGen; (2) DART; (3) E2ENLG; (4) WebNLG

2. **Translation**: Translation involves converting text from one language to another, maintaining the original meaning and nuances. We use the following datasets: (1) En-Fr from WMT'14; (2) En-De, En-Tr, En-Ru, En-Fi, En-Ro from WMT'16; (3) En-Es from Paracrawl.

3. **Commonsense Reasoning**: This involves assessing the ability to apply physical or scientific principles alongside common sense in reasoning tasks. We use the following datasets: (1) COPA; (2) HellaSwag; (3) PiQA; (4) StoryCloze.

4. **Sentiment Analysis**: A fundamental task in natural language processing (NLP) that determines the sentiment polarity (positive or negative) of a given text. We use the following datasets: (1) IMDB; (2) Sentiment140; (3) SST-2; (4) Yelp.

5. **Paraphrase Detection**: This task requires models to ascertain whether two sentences convey the same meaning, indicating semantic equivalence. We use the following datasets: (1) MRPC; (2) QQP; (3) Paws Wiki.

6. **Coreference Resolution**: Involves identifying instances within a text that refer to the same entity, demonstrating an understanding

Schemes	General	Medical	Law	Code	Math	Avg	%Param	#A	#B
Base*	38.88	35.98	33.51	20.34	10.38	27.82	-	-	-
Full*	49.91	46.78	46.08	32.93	25.70	40.28	100	-	-
Prompt Tuning*	39.91	37.59	35.02	21.55	13.18	29.45	0.001	-	-
P-Tuning*	41.11	39.81	36.72	21.13	15.56	30.87	0.193	-	-
Prefix Tuning*	41.78	40.28	36.54	22.56	16.89	31.61	0.077	-	-
IA^3*	40.45	37.12	35.25	23.17	13.98	29.99	0.009	-	-
LoRA($r = 8$)	43.44	41.18	37.95	22.82	18.72	32.82	0.062	1	1
AdaLoRA*($r = 8$)	44.32	42.83	39.36	23.78	19.51	33.96	0.093	1	1
LoRA($r = 16$)	45.12	43.22	40.24	25.22	20.14	34.79	0.124	1	1
HydraLoRA($r = 8$)	46.89	45.21	42.88	27.43	22.27	36.94	0.124	1	3
R-LoRA($r = 8$)	47.02	45.54	43.23	27.27	22.12	**37.04**	0.124	1	3

Table 6: Comparison of different training schemes on single task. * indicates results from (Tian et al., 2024)

Schemes	Task1	2	3	4	5	Avg
R-LoRA	**91.74**	**81.50**	**77.60**	**67.80**	49.30	**73.59**
w/o MD	91.40	81.20	77.10	66.10	49.10	72.98
w/o MI	91.20	80.80	77.50	66.20	**49.40**	73.02
HydraLoRA	90.97	80.30	77.20	65.80	49.20	72.69

Table 7: Results of Ablation Studies on Qwen2.5-0.5B with Different Schemes Across Various Tasks. The table compares R-LoRA with its ablated versions (without Multi-Head Dropout/MD and without Multi-Head Random Initialization/MI) against HydraLoRA across five tasks.

of textual context. We use the following datasets: (1) DPR; (2) WSC273.

7. **Reading Comprehension**: Assesses the capability to derive answers to questions from a provided text containing relevant information. We use the following datasets: (1) BoolQ; (2) DROP; (3) MultiRC; (4) OBQA; (5) SQuADv1; (6) SQuADv2.

8. **Reading Comprehension with Commonsense**: Merges traditional reading comprehension skills with commonsense reasoning, requiring understanding beyond the explicit text. We use the following datasets: (1) CosmosQA; (2) ReCoRD.

9. **Natural Language Inference**: Focuses on deducing the relationship between two sentences, determining if the second sentence logically follows from, contradicts, or is unrelated to the first sentence. We use the following datasets: (1) ANLI; (2) CB; (3) MNLI; (4) QNLI; (5) SNLI; (6) WNLI; (7) RTE.

10. **Closed-Book Question Answering**: This task challenges models to answer questions about general knowledge without direct access to external information sources. We use the following datasets: (1) ARC; (2) NQ; (3) TriviaQA.

B.4 Ablation Study

Due to limited computational resources, we selected a subset of the dataset for training and testing. **Five tasks for Smaller model Qwen2.5-0.5B in Appendix A.3**:

- Task 1: Sentiment Analysis (SST2)
- Task 2: Paraphrase Detection (QQP)
- Task 3: Natural Language Inference (QNLI)
- Task 4: Physical Commonsense Reasoning (PiQA)
- Task 5: Commonsense Reasoning (SiQA)

Eight tasks:

- Task 1: Sentiment Analysis (SST2)
- Task 2: Paraphrase Detection (QQP)
- Task 3: Natural Language Inference (MNLI + QNLI)
- Task 4: Reading Comprehension (BoolQ + OBQA)

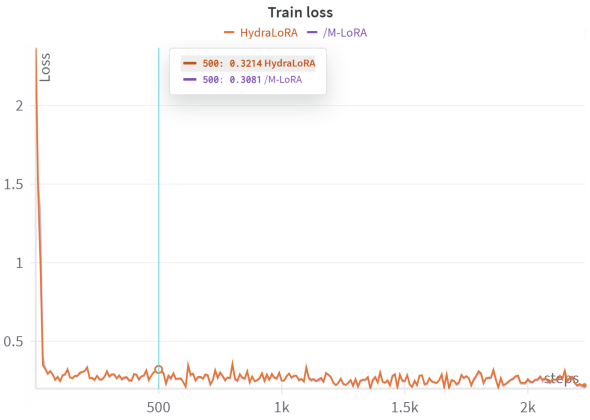

Figure 7: Training loss curves of HydraLoRA and R-LoRA. The loss of R-LoRA remains lower throughout the entire training process.

- Task 5: Commonsense Reasoning (PiQA + SiQA)
- Task 6: Reading Comprehension with Commonsense (CosmosQA)
- Task 7: Coreference Resolution (Winogrande)
- Task 8: Closed-Book Question Answering (ARC)

C Implementation Details

The hyperparameters used for training are as follows: a learning rate of 0.0002, "lora_alpha"=32, and trainable LoRA components including "gate_proj", "down_proj", and "up_proj". A dropout rate of 0.2 was applied to the LoRA, with a warmup ratio of 0.03. Mixed-precision training was enabled using bfloat16, and the learning rate scheduler was set to cosine annealing. The model was trained for 1 epoch on NVIDIA 4090 GPU. In Setting 1, the rank of LoRA was set to 10 for LoRA*2 to match the total number of trainable parameters in R-LoRA, while the rank for others was set to 4.

D Baselines

1. **Prompt Tuning**: This method adds task-specific prompts to the input. These prompt parameters are updated independently while the pretrained model parameters remain frozen.
2. **P-Tuning**: This method incorporates trainable prompt embeddings into the input, optimized by a prompt encoder to automatically discover effective prompts, removing the need for manual design. Prompt tokens can be placed anywhere in the input sequence, and anchor tokens are introduced to enhance performance.
3. **Prefix Tuning**: This method prefixes a series of task-specific vectors to the input sequence. These prefix parameters can be learned while keeping the pretrained model frozen. The prefix parameters are inserted into all layers of the model.
4. IA^3: This method enhances efficiency by infusing learned vectors into transformer architectures, drastically reducing the number of trainable parameters.
5. **AdaLoRA**: Unlike LoRA, which distributes parameters evenly across all modules, AdaLoRA optimizes the number of trainable parameters assigned to weight matrices and layers. More parameters are allocated to important weight matrices and layers, while less important ones receive fewer parameters.
6. **LoraHub** randomly aggregates 20 LoRAs for new downstream tasks. It employs a black-box optimization technique to determine the weight of each LoRA, eliminating the need for gradient calculations of the large model. This involves parameter-level weighted averaging.
7. **LoRA MoE**. A collection of n parameterized experts, denoted as $E_1, \ldots, E_n$, is orchestrated by a router network R. $E_i = B_i A_i$. Router network features a dense layer with adjustable weights W_R from $\mathbb{R}^{d_m \times n}$. A softmax function then processes an intermediate token representation x, yielding gating scores $s_1, \ldots, s_n$ that determine the weighted contribution of each expert's output:

$$s_i = R(x)_i = \text{softmax}(Top(W_R^T x, K)) \tag{5}$$

Subsequently, the overall output y is synthesized by aggregating the Top-K experts' outputs, each modulated by its respective gating score:

$$y = \sum_{i=1}^{n} s_i \cdot E_i(x) \quad \text{(MoE)} \tag{6}$$

This results in a dynamic allocation of the model's capacity, enabling specialized processing by experts as directed by the router's gating mechanism.

8. **HydraLoRA** uses a shared matrix A and multiple matrices $B_1, \ldots, B_n$. The shared matrix A is used to project the input vector x into a lower-dimensional space, while each matrix B_i is used to modulate the output of the corresponding expert E_i. The overall output y is synthesized by aggregating the experts' outputs, each modulated by its respective gating score:

$$y = \sum_{i=1}^{n} s_i \cdot (B_i \cdot A \cdot x) \tag{7}$$

This approach allows for efficient parameterization and specialization of the model's capacity, leveraging the shared matrix A for common transformations and the individual matrices B_i for task-specific adjustments.