R-LoRA: Random Initialization of Multi-Head LoRA for Multi-Task Learning

Anonymous ACL submission

Abstract

Fine-tuning large language models (LLMs) is prohibitively expensive in terms of computational and memory costs. Low-rank Adaptation (LoRA), as one of the most popular parameter-efficient fine-tuning (PEFT) methods, offers a cost-effective alternative by approximating the model changes $\Delta W \in \mathbb{R}^{m \times n}$ through the product of down-projection matrix $A \in \mathbb{R}^{m \times r}$ and head matrix $B \in \mathbb{R}^{r \times n}$, where $r \ll \min(m, n)$. 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 Random Initialization and Multi-Head Dropout, enabling more efficient learning of task-specific features while maintaining shared knowledge representation. Extensive experiments demonstrate that R-LoRA is better at capturing task-specific knowledge, thereby improving performance in multi-task scenarios.

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, such as prefix-tuning(Li and Liang, 2021), p-tuning(Liu et al., 2024c), IA^3(Liu et al., 2022) and Low-rank adaption(LoRA)(Hu et al., 2021), have been widely studied. 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 pa-

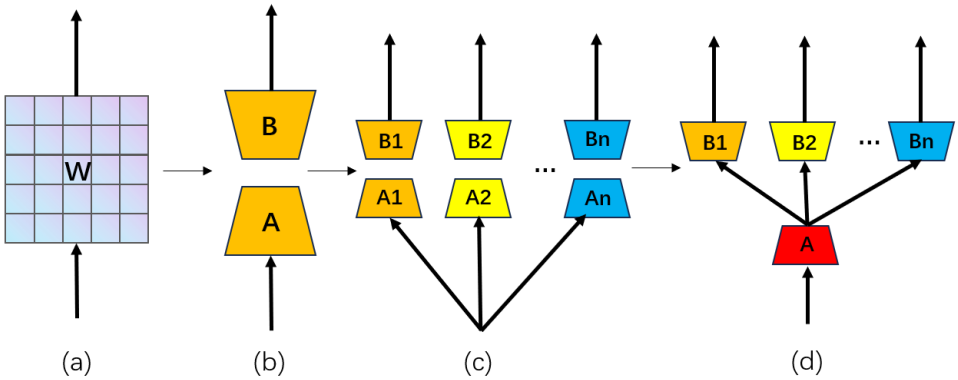

Figure 1: Training architecture comparison. (a) Full parameter fine-tuning; (b) Vanilla LoRA; (c) Multi-Adapter architecture; (d) Multi-Head/Asymmetric architecture.

rameters of the down-projection matrices A are relatively consistent, while the differences between the head matrices B are more pronounced, which 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. The mathematical formalization of HydraLoRA is detailed in Section 2.2. In this work, we propose R-LoRA, which adopts HydraLoRA's asymmetric architecture, explicitly defining it as a Multi-Head structure, and introduce Multi-Head randomization to improve LLMs' performance on multi-task learning. 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 and slowing convergence speed. This is due to zero initialization of head matrices B, leading to similar update directions. To address this, we use multi-head randomization in R-LoRA, combining random initialization and multi-head dropout to diversify starting points and inputs, thereby improving task-specific learning. This approach enables LLMs to better learn task-specific knowledge by breaking the symmetry of initial parameters and diversifying optimization trajectories. Extensive experiments demonstrate the effectiveness of our proposed method. R-LoRA achieves significant improvements in multi-task scenarios while delivering modest gains in single-task contexts, showcasing its adaptability across a variety of tasks.

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. 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:

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

Normally matrix B is initialized with zeroes and matrix A is initialized with Kaiming Uniform (He et al., 2015) to ensure that the initial outputs are 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

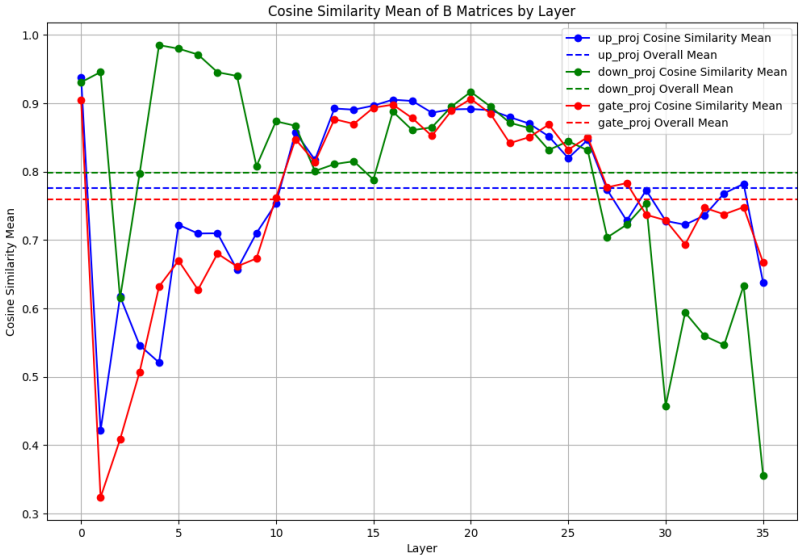

Figure 2: Cosine similarity among head matrices. "Overall mean" represents the average similarity across all layers.

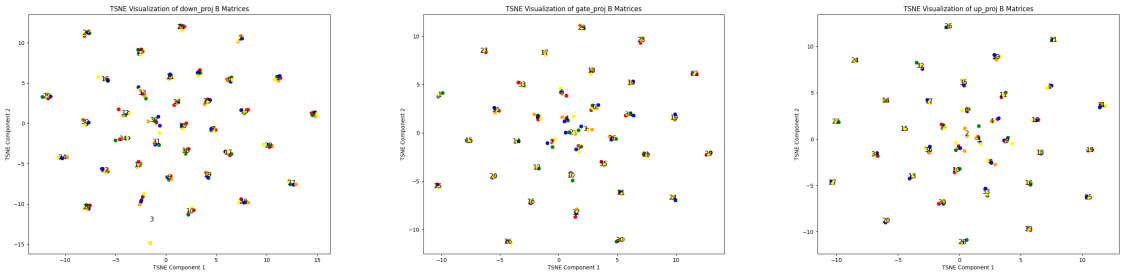

Figure 3: T-SNE analysis of head matrices

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. LoSparse (Li et al., 2023a) incorporates LoRA to prevent pruning from eliminating too many expressive neurons. 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 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:

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

Normal routing matrix is initialized with Kaiming Uniform (He et al., 2015). R-LoRA retains the

same architecture as HydraLoRA, ensuring consistency in the routing mechanism and weight computation.

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. 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.

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-Base (Qwen Team, 2024) with HydraLoRA (Tian et al., 2024) on five different 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). 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.

As shown in Figure 2 and Figure 3, 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 and the shared A matrix. After receiving the outputs from shared down-projection matrix A, the outputs of the head matrices are highly similar in the early stages of training, leading to highly similar update directions during gradient updates. The differences in the updates of the head matrices arise solely from the Kaiming uniform initialization (He et al., 2015) of the router matrix, which is insufficient.

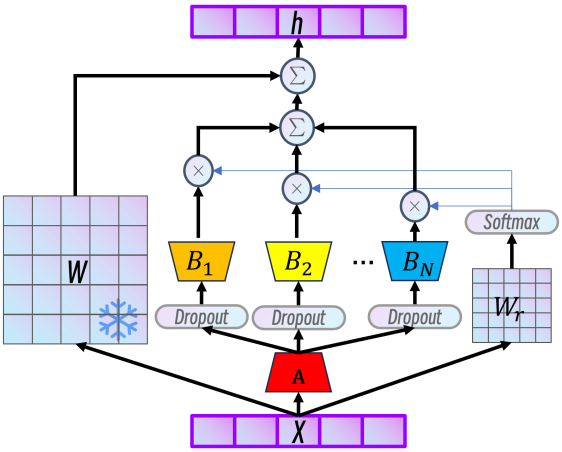

Figure 4: Overview of the R-LoRA.

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

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 4

Reserach Objective: *To exploit randomization to differentiate the head matrices, thereby facilitating the convergence 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. We hypothesize that the down-projection matrix A is more inclined to learn task-agnostic knowledge, capturing general features applicable across tasks, while the head matrices tend to specialize in task-specific knowledge, enabling the model to differentiate and adapt to

Method	A Initialization	B Initialization
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.

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 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 4. Our architecture is similar to HydraLoRA (Tian et al., 2024), but it introduces multi-head dropout. The input, after being processed by the down-projection matrix A, obtains a task-agnostic representation. Multi-head dropout diversifies this representation, enabling the model to learn task-specific knowledge from multiple perspectives, enhancing both generalization and task adaptability.

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 shown in Table 1, we utilize non-zero initialization for the head matrices to provide them with distinct starting points during training, thereby encouraging them to converge to different positions. However, multi-head dropout introduces greater variance to the inputs of the head matrices. While this variance aids the model in learning diverse knowledge, it may also lead to scaling instability. 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 matrix in order to ensure scale stability. The γ is a hyperparameter, and we follow the setting of (Wang et al., 2024) by setting it to 64. When the head matrices are initialized with non-zero values, the $\Delta\mathbf{W}$ is no longer zero. To ensure that the initial outputs are approximately consistent with the pre-trained model and reduce introducing disturbances, we subtract the initial $\Delta\mathbf{W}$ from the original parameter matrix $\mathbf{W}$. It can be formulated as:

$$W = W - \frac{1}{N}\sum_{i=1}^{N} \cdot B_i A \tag{4}$$

5 Experiment

In this section, we validate the superiority of R-LoRA across various models and settings. First, we followed the settings of (Tian et al., 2024) and conducted experiments on the LLaMA-2 model (Touvron et al., 2023), evaluating both single-task and multi-task scenarios. Subsequently, we tested the performance of R-LoRA under different multi-task settings on the new Qwen2.5 (Qwen Team, 2024). The model sizes range from 0.5B to 13B. Through an extensive ablation study, we demonstrate the effectiveness of the multi-head randomization in R-LoRA.

5.1 Experiment Setting

Model: In the single-task setting, we use LLaMA2-7B, while in the multi-task setting, we additionally incorporated LLaMA2-13B. In the ablation study, we use Qwen2.5-0.5B and Qwen2.5-3B models.
Dataset & Benchmarks:
Single-task:

- **General:** We fine-tune the model using the general instruction tuning dataset Databricks-Dolly-15k (Conover et al., 2023), which focuses on generic language capabilities. The performance is then evaluated using the MMLU benchmark (Hendrycks et al., 2020).
- **Medical:** We fine-tune the model using GenMedGPT and the Clinic-10k dataset from ChatDoctor (Li et al., 2023b), targeting medical applications. The model's performance on medical tasks is assessed using MMLU.
- **Law:** Fine-tuning is conducted using two legal instruction datasets, Lawyer-Instruct (Alignment-Lab-AI, 2023) and US-Terms (Chalkidis et al., 2023), and the model is evaluated on legal tasks using MMLU.

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($r = 8$)	47.12	45.28	43.28	27.43	22.27	37.08	0.124	1	3
R-LoRA($r = 8$)	47.79	45.31	43.22	27.27	22.12	**37.13**	0.124	1	3

Table 2: Comparison of different training schemes on single task. * indicates results from (Tian et al., 2024)

Metrics	Base	LoRA	LoRAHub*	LoRA MoE*	HydraLoRA	R-LoRA
7B	31.6	37.2	39.7	40.3	41.8	**42.3**
13B	38.4	40.9	41.9	43.7	44.7	**45.4**
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 task. * indicates results from (Tian et al., 2024)

- **Code:** We fine-tune the model using the CodeAlpaca (Chaudhary, 2023) for code generation tasks, and the evaluation is conducted using the HumanEval (Chen et al., 2021).
- **Math:** The model is fine-tuned on the training set of GSM8K (Cobbe et al., 2021) to enhance its mathematical reasoning capabilities and is evaluated on the corresponding test set

Multi-task: We fine-tune the model using a subset of the Flanv2 dataset (Brown, 2020) that includes tasks from both Natural Language Understanding (NLU) and Natural Language Generation (NLG), grouped into 10 distinct task clusters. The model's performance is evaluated using the Big-Bench Hard (BBH) benchmark. More Details of the datasets will be provided in the appendix A.

Baseline: First, 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). Second, we extend the comparison by evaluating R-LoRA against other weighted averaging methods across multiple datasets: 1) Lorahub (Huang et al., 2023), which utilizes black-box optimization to learn weights for 20 randomly selected LoRAs for new tasks, applying weighted averaging without the need for gradient calculations; 2) LoRA MoE (Liu et al., 2024a), which combines lightweight experts (LoRA) with a Mixture of Experts (MoE) architecture for high efficiency, enabling generalization to new tasks without prior knowledge; 3) HydraLoRA (Tian et al., 2024), which employs Multi-Head structure in conjunction with MoE to achieve a balance between parameter efficiency and training effectiveness.

5.2 Performance

5.2.1 Performance of R-LoRA on Single Task

As shown in Table 2, in the single-task setting, where the knowledge and text format of the data are relatively homogeneous, multi-head randomization does not yield significant performance improvements. Nevertheless, R-LoRA achieves performance on par with HydraLoRA, demonstrating that the multi-head randomization mechanism preserves learning effectiveness while maintaining stability for single-task scenarios. This highlights R-LoRA's robustness and adaptability, even in settings where its full potential may not be fully utilized.

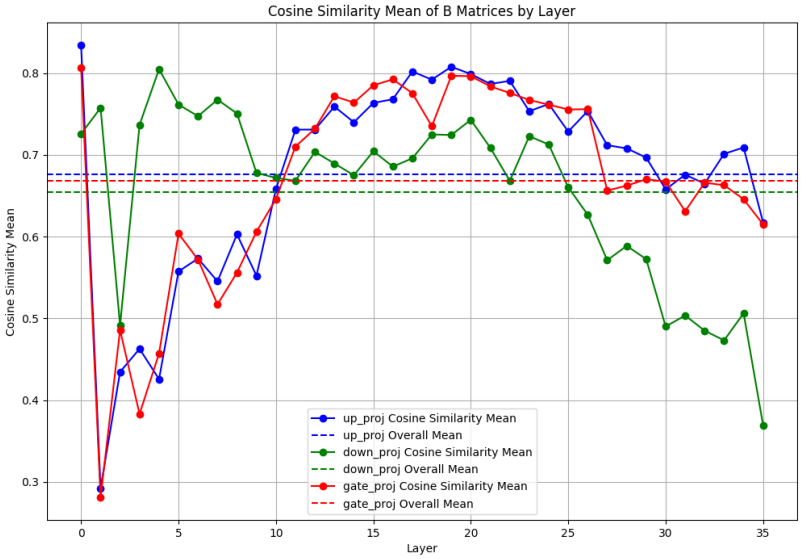

Figure 5: Cosine similarity among head matrices in R-LoRA. "Overall mean" represents the average similarity across all layers.

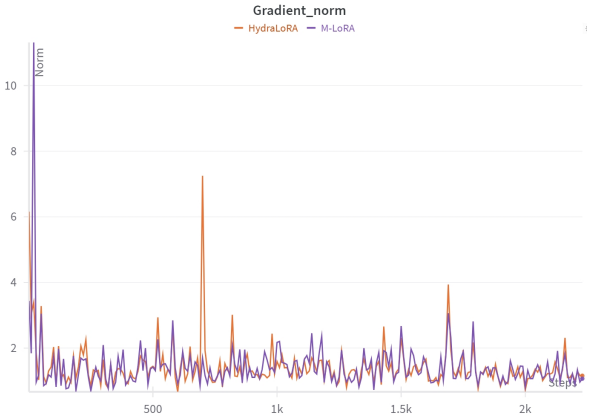

Figure 6: Gradient norms of HydraLoRA and R-LoRA.

5.2.2 Performance of R-LoRA on Multi-Tasks

The evaluation across diverse tasks, as shown in Table 3, demonstrates that R-LoRA, building upon the foundation of HydraLoRA, consistently outperforms all other schemes. By introducing multi-head dropout and random initialization for the head matrices, R-LoRA further enhances the model's stability and adaptability. The performance gains of R-LoRA, rooted in these multi-head randomization techniques, surpass those of both conventional PEFT methodologies and HydraLoRA.

5.3 Parameter Analysis

Reserach 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 5, 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. T-SNE analysis will be shown in appendix C

5.4 Training Process

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

As illustrated in Figure 6, R-LoRA benefits from multi-head randomization, exhibiting significantly 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 demonstrates greater training stability than HydraLoRA, 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.

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

Table 4: Results of Ablation Studies on Qwen2.5-0.5B with Different Schemes Across Various Tasks. The table compares the performance of HydraLoRA, +MT (HydraLoRA with Multi-Head Dropout), +Init (Multi-Head Random Initialization), and R-LoRA across five tasks.

Schemes	Task1	2	3	4	5	6	7	8	Avg
Qwen2.5-0.5B									
HydraLoRA	87.5	77	73.5	63.5	52.5	52	54.5	52.5	64.13
+MT	88	78.5	74.5	66.5	54.75	53.5	58.5	52.5	66.22
+Init	87.5	79.75	75.75	65.75	53	58	56	55	66.31
R-LoRA	89	82.5	76.5	66.5	55	57	55.5	56.5	**67.31**
Qwen2.5-3B									
HydraLoRA	95.5	84	83.5	83.5	71.25	72	83.5	88	82.66
+MT	96	83.5	83.5	84.5	71.75	72	83.5	88	82.84
+Init	96.5	84	83.5	84.75	71.5	73	85	88.5	83.34
R-LoRA	96.5	83.5	85	86.5	72.75	73.5	86	89	**84.09**

Table 5: Results of Ablation Studies on Qwen2.5-0.5B and Qwen2.5-3B Models with Different Schemes Across Various Tasks. The table compares the performance of HydraLoRA, +MT (HydraLoRA with Multi-Head Dropout), +Init (Multi-Head Random Initialization), and R-LoRA across eight tasks.

5.5 Ablation Study

In this section, we empirically validate the effectiveness of R-LoRA's multi-head randomization components through extensive experiments. Ablation studies were conducted on two models, Qwen2.5-0.5B and Qwen2.5-3B, under two task settings: 5-task and 8-task configurations. For the 5-task setting, models were fine-tuned on 5 datasets spanning two categories (NLU and commonsense reasoning). For the 8-task setting, models were trained on 11 datasets across 8 categories, with all models evaluated on their respective test sets. The experimental results are presented in Table 4 and Table 5. Details of the datasets will be provided in the appendix A.3.

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. Random initialization assigns unique weights to each head matrix, enabling the capture of task-specific patterns and reducing the risk of head convergence. Dropout diversifies inputs to the head matrices, fostering distinct learning pathways and reducing redundancy. Together, these components improve task-specific feature capture while ensuring robustness in multi-task learning.

6 Conclusion

In this work, we first investigated the multi-head structure of LoRA and analyzed the parameters of the head matrices, revealing that they remain highly similar. To address this, we proposed R-LoRA, which introduces multi-head randomization—a simple yet effective approach—to enable the model to learn knowledge from different tasks, thereby enhancing its performance in multi-task scenarios. This method not only improves the model's generalization capabilities but also supports its adaptability across diverse tasks. Extensive experiments have validated the superiority of R-LoRA. Parameter analysis demonstrates that multi-head randomization effectively differentiates the head matrices, enabling them to learn knowledge from distinct tasks. This capability significantly enhances the model's performance in multi-task scenarios, confirming the effectiveness of the proposed approach.

7 Limitation

Despite the promising results of R-LoRA, several limitations should be acknowledged. While empirical evidence supports the effectiveness of multi-head randomization, a rigorous theoretical analysis of its underlying mechanisms remains absent. Additionally, multi-head random initialization does not ensure consistency with the pre-trained model's outputs, potentially introducing random disturbances. Future work could explore data-driven initialization as a promising approach to enhance the learning of task-specific knowledge by the head matrices, a direction we intend to pursue further.

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.

Alignment-Lab-AI. 2023. Lawyer-instruct dataset. https://huggingface.co/datasets/Alignment-Lab-AI/Lawyer-instruct.

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*.

Ilias Chalkidis, Nicolas Garneau, Catalina Goanta, Daniel Martin Katz, and Anders Søgaard. 2023. Lexfiles and legallama: Facilitating english multinational legal language model development. *arXiv preprint arXiv:2305.07507*.

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.

Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation. https://github.com/sahil280114/codealpaca.

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

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*.

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. *Company Blog of Databricks*.

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.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

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*.

Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2023a. Losparse: Structured compression of large language models based on low-rank and sparse approximation. In *International Conference on Machine Learning*, pages 20336–20350. PMLR.

Yunxiang Li, Zihan Li, Kai Zhang, Ruilong Dan, Steve Jiang, and You Zhang. 2023b. Chatdoctor: A medical chat model fine-tuned on a large language model meta-ai (llama) using medical domain knowledge. *Cureus*, 15(6).

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 Datasets

A.1 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".

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`.

A.2 Multi-task

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 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.

A.3 Ablation Study

Due to limited computational resources, we selected a subset of the dataset for training and testing.
Five tasks:

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

B 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.

C More Results

The T-SNE analysis of R-LoRA has been shown in Figure 7, 8, 9

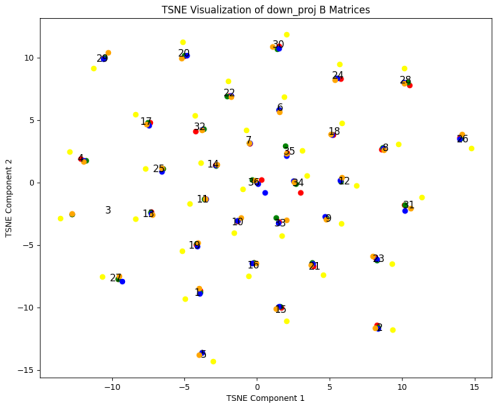

Figure 7: T-SNE analysis of head matrices(down_proj) in R-LoRA.

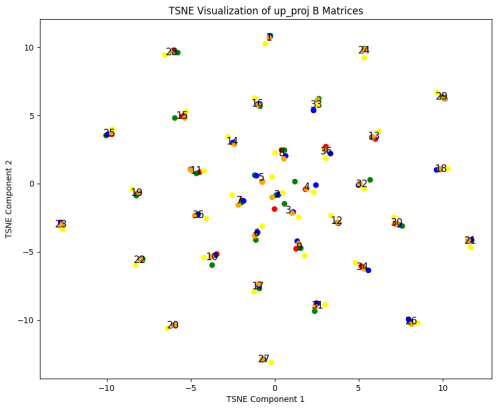

Figure 8: T-SNE analysis of head matrices(up_proj) in R-LoRA.

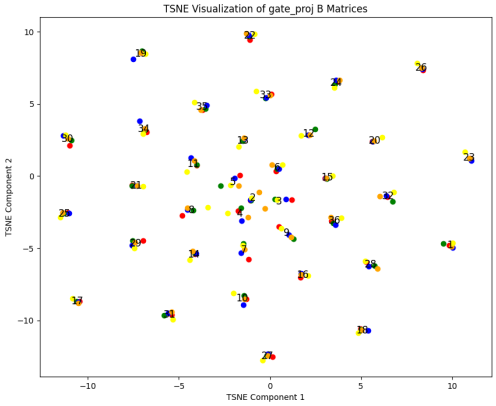

Figure 9: T-SNE analysis of head matrices(gate_proj) in R-LoRA.

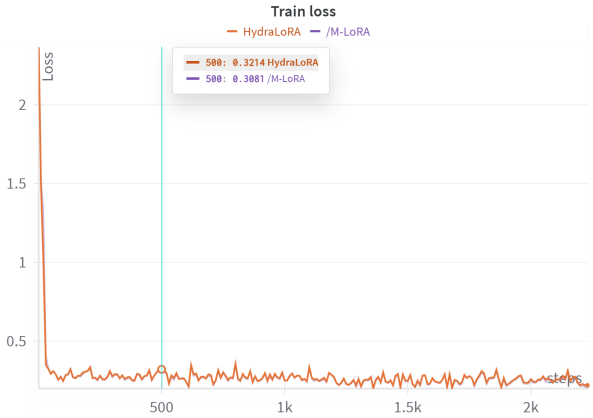

Figure 10: Training loss curves of HydraLoRA and R-LoRA. The loss of M-LoRA remains lower throughout the entire training process..