Increasing the Rank: Revisiting the LoRA Architechture in Multi-Task Learning

Anonymous ACL submission

Abstract

Fine-tuning large language models (LLMs) is computationally expensive, and Low-Rank Adaptation (LoRA) offers a cost-effective alternative by approximating weight updates with low-rank matrices. In multi-task learning (MTL) scenarios, while recent works have introduced multi-head LoRA variants to capture task-specific knowledge across different tasks, we observe a high degree of similarity among head matrices, questioning the necessity of such structural complexity for multi-task generalization. In this work, we propose R-LoRA+, a simplified but competitive multi-head LoRA. We highlight that increasing the rank of standard LoRA suffices to match or even surpass the performance of methods with multi-adapter or multi-head, suggesting that structural diversification may not be necessary for multi-task generalization. Furthermore, we find that explicitly encouraging shared representation learning leads to more effective adaptation under parameter-efficient fine-tuning. Experimental results confirm that focusing on shared knowledge across tasks improves multi-task generalization while preserving the deployment-friendly nature of LoRA.

1 Introduction

In recent years, large language models (LLMs) have demonstrated unprecedented performance across a wide range of natural language processing (NLP) tasks (Brown, 2020; Zhao et al., 2023; Chang et al., 2024b). Their remarkable capabilities in language understanding and generation have attracted widespread attention from both academia and industry. Despite their strong generalization abilities, LLMs often require further adaptation to align with domain-specific requirements or to incorporate updated knowledge (Agiza et al., 2024; Xin et al., 2024).

Supervised fine-tuning (SFT) plays a critical role in aligning LLMs with human instructions by training the model on a small but high-quality set of labeled examples (Hu et al., 2021; Xia et al., 2024). However, full fine-tuning (FT), which involves updating all model parameters, poses significant challenges in terms of computational efficiency and memory consumption due to the enormous scale of modern LLMs (Mao et al., 2025).

To address the high hardware demands of adapting LLMs, parameter-efficient fine-tuning (PEFT) methods have been proposed (Han et al., 2024; Chang et al., 2024a). These approaches significantly reduce VRAM consumption, especially for optimizer states, by updating only a small subset of parameters while keeping the majority of the model weights frozen. A variety of PEFT techniques have been extensively studied, including prefix tuning (Li and Liang, 2021), prompt tuning(Liu et al., 2024c), adapter-based methods (Liu et al., 2022), and low-rank adaptation (LoRA) (Hu et al., 2021), among others.

Among various parameter-efficient methods, LoRA has emerged as the most widely adopted alternative to full fine-tuning. Instead of directly updating the original weight matrices, LoRA introduces low-rank matrices to approximate the parameter updates via matrix decomposition. During inference, the original weights and adapted matrices are combined to produce the final model outputs. In practice, LLMs are often fine-tuned on data from multiple domains to perform a diverse set of tasks, naturally aligning with the multi-task learning (MTL) paradigm.

Recent advances in LoRA have introduced multi-adapter architectures to enhance multi-task learning, with notable variants including Multi-LoRA (Wang et al., 2023), LoRA-MoE (Dou et al., 2023), and MoeLoRA (Liu et al., 2024a). Liu et al. (2025) refers to this general framework as the **Multi-Adapter** LoRA architecture, which consists of multiple down-projection matrices $\mathbf{A}$ and their corresponding head matrices $\mathbf{B}$. This design

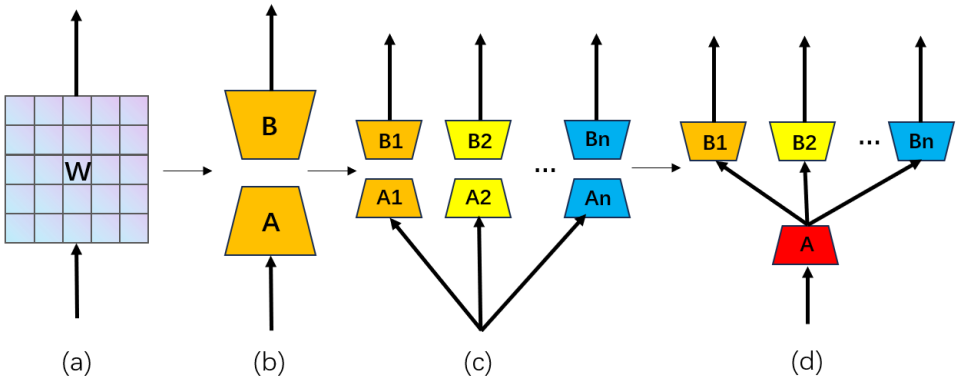

Figure 1: Training architecture comparison. (a) Full parameter fine-tuning; (b) Vanilla LoRA; (c) Multi-Adapter architecture; (d) Multi-Head/Asymmetric architecture.

enables task-specific adaptation by allowing each task to utilize a distinct set of adapter parameters.

Among these methods, LoRA-MoE and MoeLoRA further improve performance by incorporating a Mixture-of-Experts (MoE) mechanism to dynamically aggregate outputs from different adapters. Tian et al. (2024) observes that in the LoRA framework, the down-projection matrices $\mathbf{A}$ tend to be highly similar across tasks, whereas the head matrices $\mathbf{B}$ exhibit greater variation, indicating that they are more responsible for capturing task-specific knowledge. Motivated by this observation, HydraLoRA (Tian et al., 2024) proposes an asymmetric architecture that shares a single down-projection matrix $\mathbf{A}$ across all tasks while maintaining multiple task-specific $\mathbf{B}$ matrices. In addition, HydraLoRA employs an MoE-based routing mechanism to combine the outputs of the head matrices.

Building on this line of work, R-LoRA (Liu et al., 2025) explicitly interprets the asymmetric architecture as a **Multi-Head** LoRA structure and reveals that head matrices often exhibit high similarity, leading to redundancy across heads. To mitigate this issue, R-LoRA introduces multi-head randomization, encouraging each head to learn diverse and task-specific representations, further improving model expressiveness while reducing GPU memory consumption and computational cost. The mathematical formulation of the multi-head structure is detailed in Section 2.2, and the architectural differences among the aforementioned approaches are illustrated in Figure 1.

However, multi-head LoRA architectures such as HydraLoRA rely on dynamic routing mechanisms that prevent adapter weights from being merged into the base model, resulting in additional inference overhead. In this work, we propose **R-LoRA+**, a simplified but competitive multi-head LoRA. Our analysis reveals that the Multi-Adapter and Multi-Head structure may not be essential for effective adaptation, as a simplified design can yield superior or comparable results to advanced multi-head variants. Moreover, we highlight that increasing the rank of standard LoRA suffices to match or even surpass the performance of methods with multi-adapter or multi-head. This observation motivates a shift in focus toward learning task-shared knowledge, rather than enforcing explicit task-specific specialization. Building on this insight, we propose **Align-LoRA**, which further improves multi-task generalization by aligning the representations of the down-projection matrix $\mathbf{A}$ through MK-MMD (Sejdinovic et al., 2013; Gretton et al., 2012) regularization.

Our key contributions are:

- *We propose **R-LoRA+**, a simplified but competitive multi-head LoRA, challenging the need for complex routing mechanisms.*

- *We highlight that **increasing the rank** of standard LoRA suffices to match or exceed the performance of multi-head variants, suggesting that structural complexity is not essential.*

- *We introduce **Align-LoRA**, which improves multi-task generalization by aligning task-shared representations without introducing additional parameters.*

2 Related Works

2.1 LoRA

Current LLMs typically adopt a decoder-only architecture, consisting of stacked transformer blocks (Zhao et al., 2023). Each block contains two core components with residual connections: a

multi-head self-attention (MHA) layer and a feed-forward network (FFN) (Vaswani, 2017). Both layers rely on dense learnable weight matrices $\mathbf{W}$ for feature transformation.

To efficiently adapt LLMs to specific tasks or domains under resource constraints, LoRA (Hu et al., 2021) offers an effective solution. Inspired by the hypothesis that the intrinsic dimensionality of parameter updates in LLMs is low, LoRA approximates the weight update $\Delta\mathbf{W}$ using two low-rank matrices $\mathbf{A} \in \mathbb{R}^{r\times n}$ and $\mathbf{B} \in \mathbb{R}^{m\times r}$, where $\mathbf{W} \in \mathbb{R}^{m\times n}$ is the original weight matrix. The rank r is chosen to be significantly smaller than $\min(m, n)$, reducing the number of trainable parameters from $\mathcal{O}(mn)$ to $\mathcal{O}(r(m + n))$. This results in the updated weight matrix being expressed as $\mathbf{W} + \mathbf{BA}$. Given an input x, the corresponding output h becomes:

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

where $\Delta\mathbf{W} = \mathbf{BA}$ denotes the low-rank update.

In practice, matrix $\mathbf{B}$ is initialized with zeros to ensure that no random perturbations are introduced at the beginning of training, while $\mathbf{A}$ is initialized using Kaiming Uniform (He et al., 2015). This initialization strategy ensures that the initial outputs remain consistent with the pre-trained model.

Following the original LoRA framework, several works have proposed improvements to enhance adaptability and efficiency. AdaLoRA (Zhang et al., 2023) dynamically adjusts the rank during training for layer-wise optimization. DeltaLoRA (Zi et al., 2023) and DoRA (Liu et al., 2024b) refine model updates by decomposing weight changes into magnitude or direction components. PiSSA (Meng et al., 2025) and LoRA-GA (Wang et al., 2024) improve convergence through better initialization strategies. NLoRA (Guo et al., 2025) further enhances expressiveness and stability by decomposing the parameter matrix into three components initialized via the Nyström method. These approaches highlight the importance of structural design and initialization in improving LoRA's performance within parameter-efficient fine-tuning frameworks.

2.2 Multi-Head Architecture

MTL-LoRA (Yang et al., 2024) and HydraLoRA (Tian et al., 2024) are among the first to introduce the multi-head architecture into LoRA-based parameter-efficient fine-tuning. This architecture consists of a shared down-projection matrix $\mathbf{A}$ and multiple distinct head matrices $\mathbf{B}_i$, enabling both task-specific adaptation and knowledge sharing across tasks.

As illustrated in Figure 1, this design effectively separates task-specific components while preserving shared representations across different tasks. The overall weight update in the Multi-Head architecture can be expressed as:

$$\mathbf{W} + \Delta\mathbf{W} = \mathbf{W} + \sum_{i=1}^{N} \omega_i \cdot \mathbf{B}_i\mathbf{A}, \quad (2)$$

where N is the number of heads, and ω_i denotes the weight assigned to the i-th adapter output.

In MTL-LoRA and HydraLoRA, the weights ω_i are computed using a routing mechanism based on the input representation. Specifically, they employ a learnable routing matrix $\mathbf{W}_r$ followed by a softmax function:

$$\boldsymbol{\omega} = \mathrm{Softmax}(\mathbf{W}_r\mathbf{x}), \quad (3)$$

where $\mathbf{x}$ is the input token representation and $\boldsymbol{\omega}$ represents the normalized weights assigned to each head.

Liu et al. (2025) observe that head matrices in multi-head LoRA often exhibit high similarity, leading to redundancy. To address this, R-LoRA introduces multi-head randomization to encourage diverse task-specific learning, achieving better performance with reduced computational and memory overhead.

2.3 Maximum Mean Discrepancy (MMD)

Maximum Mean Discrepancy (MMD)(Sejdinovic et al., 2013) is a kernel-based statistical measure for quantifying the difference between two probability distributions. Given a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_k$ with a characteristic kernel k, the squared MMD between representation p and q is defined as:

$$\mathrm{MMD}^2(p, q) = \|\mu_k(p) - \mu_k(q)\|^2_{\mathcal{H}_k}, \quad (4)$$

where $\mu_k(p) = \mathbb{E}_{\mathbf{x}\sim p}[\phi(\mathbf{x})]$ and $\mu_k(q) = \mathbb{E}_{\mathbf{y}\sim q}[\phi(\mathbf{y})]$ are the mean embeddings of p and q in $\mathcal{H}_k$, and $\phi(\cdot)$ denotes the feature mapping induced by kernel k.

A key advantage of MMD is its ability to capture distributional differences in feature spaces without requiring explicit density estimation. However, its performance heavily depends on the choice of kernel. To address this limitation, the Multiple Kernel

MMD (MK-MMD)(Gretton et al., 2012) extends MMD by combining multiple kernels adaptively:

$$\text{MK-MMD}^2(p, q) = \sum_{k \in \mathcal{K}} \|\mu_k(p) - \mu_k(q)\|^2_{\mathcal{H}_k}, \tag{5}$$

where $\mathcal{K}$ is a predefined set of kernels. This variant enhances robustness by combining multiple kernels, allowing the metric to capture multi-scale distributional discrepancies in the reproducing kernel Hilbert space (RKHS).

In the context of transfer learning and domain adaptation, MMD has been widely used as a criterion for aligning feature distributions between source and target domains(Ben-David et al., 2006; Pan et al., 2010). The core idea is to minimize the MMD distance between activations from different domains, encouraging the model to learn domain-invariant representations that generalize well across tasks.

For example, in unsupervised domain adaptation (UDA), MMD is often applied to match the feature distributions of labeled source data and unlabeled target data, reducing domain shift and improving generalization performance. Specifically, given features from the source domain $\mathcal{D}_s = \{\mathbf{x}_i^s\}_{i=1}^{n_s}$ and the target domain $\mathcal{D}_t = \{\mathbf{x}_j^t\}_{j=1}^{n_t}$, the MMD loss is defined as:

$$\mathcal{L}_{\text{MMD}} = \left\| \frac{1}{n_s} \sum_{i=1}^{n_s} \phi(\mathbf{x}_i^s) - \frac{1}{n_t} \sum_{j=1}^{n_t} \phi(\mathbf{x}_j^t) \right\|^2, \tag{6}$$

where $\phi(\cdot)$ denotes the feature embedding function, and the objective is to minimize the distributional discrepancy between the two domains in the shared feature space.

In the context of neural networks for image classification, this MMD loss can be incorporated into the overall training objective alongside the standard classification loss (Long et al., 2015). The total loss function is typically formulated as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{cls}} + \lambda \cdot \mathcal{L}_{\text{MMD}}, \tag{7}$$

where $\mathcal{L}_{\text{cls}}$ is the cross-entropy loss on the labeled source data, and λ is a hyperparameter that balances the contribution of the MMD regularization.

Building on this principle, we propose to incorporate MMD into LoRA for multi-task learning, with a focus on its multiple kernel extension, MK-MMD. Unlike traditional applications that focus on aligning input or hidden-layer features, we apply MK-MMD to the output of the low-rank down-projection matrix $\mathbf{A}$ in LoRA, encouraging the model to learn shared, task-agnostic representations. This design improves multi-task generalization by reducing distributional discrepancies in the representation space, without introducing additional parameters.

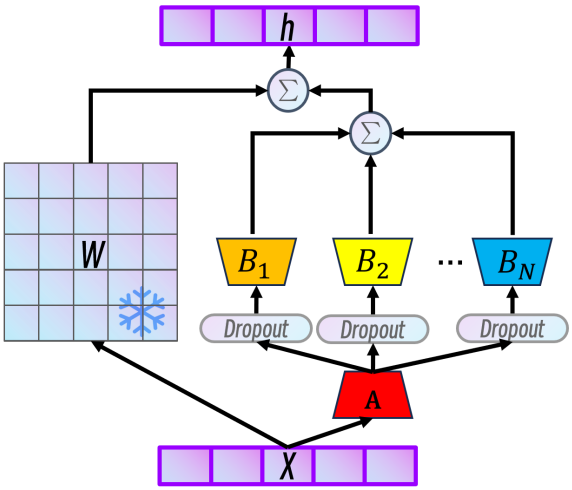

Figure 2: Overview of the R-LoRA+ framework. Our simplified variant of R-LoRA removes the dynamic routing (Router) module and instead applies a fixed average fusion mechanism to combine the outputs of all head matrices during inference.

3 Observations

3.1 Head Matrices in Multi-Head LoRA

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 R-LoRA (Liu et al., 2025) and use cosine similarity to observe the parameters of the head matrices.

In this work, we propose a simplified variant of R-LoRA, which we denote as R-LoRA+, by removing its dynamic routing (Router) module. In R-LoRA+, the outputs of the head matrices are simply averaged during training, without input-dependent weighting. An overview of the R-LoRA+ framework is illustrated in Figure 2, highlighting its structural simplicity and efficient adaptation mechanism. We fine-tune Qwen2.5-3B (Qwen Team, 2024) with HydraLoRA (Tian et al., 2024) and R-LoRA+ on five different tasks. All experimental setups for this work, including dataset descriptions, training procedures, and hyperparameter configurations, are comprehensively documented in Appendix B. To

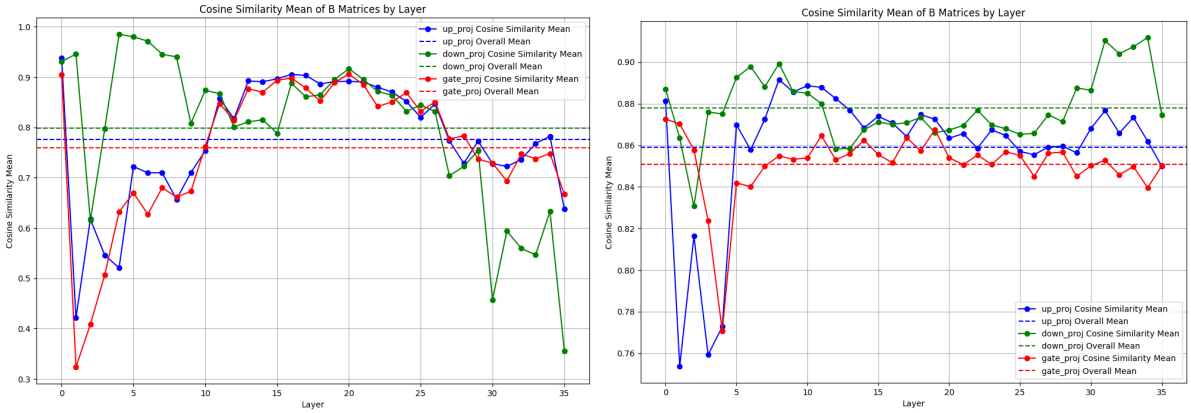

Figure 3: Cosine similarity among head matrices for HydraLoRA (left) and R-LoRA+ (right). The left plot shows that HydraLoRA maintains moderate similarity across heads, while the right plot reveals that removing the router in R-LoRA (R-LoRA+) leads to higher similarity among head matrices. "Overall mean" denotes the average similarity across all model layers.

Schemes	QNLI	PiQA	Winogrande	ARC	GSM8K	Avg	%Para
HydraLoRA	81.90	84.21	70.90	87.21	45.95	74.03	0.45
R-LoRA	82.00	85.55	71.80	87.69	46.25	74.66	0.45
R-LoRA+	**82.30**	**86.76**	**72.90**	**88.03**	**46.85**	**75.34**	**0.41**

Table 1: Comparative study of several multi-head LoRA variants across five tasks.

evaluate the parameter similarity among head matrices, we first flatten each matrix into a vector and then compute pairwise cosine similarities to construct a similarity matrix. The average value of this matrix is used as an overall measure of head matrix similarity. Additionally, we perform T-SNE analysis on all the head matrices of HydraLoRA in Figure 5 of Appendix A.

As illustrated in the left plot of Figure 3, HydraLoRA, which is a typical example of multi-head LoRA, still exhibits over 70% similarity among its head matrices. This suggests that, despite its multi-head design, HydraLoRA still learns substantial shared knowledge across heads, limiting its ability to capture task-specific features. To address this limitation, R-LoRA introduces multi-head randomization to encourage diverse knowledge learning across tasks, thereby improving multi-task performance. The analysis of R-LoRA can be refered to Figure 4 in Appendix A.

Although R-LoRA was originally designed to improve multi-task learning by reducing redundancy among head matrices, as shown in Figure 4, the right plot of Figure 3 reveals that removing its Router module leads to an even higher similarity among these matrices in R-LoRA+, suggesting that the Router plays a key role in maintaining diversity across heads. Across all adapter modules, the parameter similarity exceeds 75%, with an average of over 85%. This observation raises a natural question:

RQ 1: *How does head matrix similarity affect multi-task learning performance? In particular, can high similarity among head matrices coexist with strong generalization across tasks?*

3.2 Multi-Task Performance Comparison

To assess the impact of head matrix similarity on multi-task performance, we conduct a comparative study of several multi-head LoRA variants, including HydraLoRA, R-LoRA, and its simplified variant R-LoRA+. As shown in Table 1, we find that although R-LoRA improves multi-task performance by encouraging the model to learn diverse task-specific knowledge, R-LoRA+, which removes the Router module and shows the highest head matrix similarity among the three variants, achieves even better results. This finding goes against intuition. Notably, R-LoRA introduces multi-head randomization to promote diversity, and compared to HydraLoRA, it reduces memory consumption and computational cost despite having the identical number of trainable parameters. By further removing the Router in R-LoRA+, the number of trainable parameters is reduced even more, leading to additional gains in efficiency in terms of both

Metrics	LoRA	LoRAHub*	LoRA MoE*	HydraLoRA	R-LoRA	LoRA†
7B	37.1	39.7	40.3	41.5	**42.2**	**42.2**
13B	40.8	41.9	43.7	44.2	**45.1**	<u>44.9</u>
% Param	0.06	1.24	2.98	0.34	0.34	0.34

Table 2: Comparison of different training schemes on Llama2. LoRA† denotes the variant where the LoRA rank is increased to match the number of trainable parameters in multi-head variants. * indicates results from (Tian et al., 2024).

Metrics	LoRA^{4}	LoRA^{8}	LoRA^{9}	LoRA^{10}	HydraLoRA	R-LoRA	R-LoRA+
7B	43.21	46.66	48.18	<u>49.48</u>	49.12	**49.51**	<u>49.48</u>
Rank	4	8	9	10	4	4	4
% Param	0.10	0.20	0.22	0.25	0.25	0.25	0.22

Table 3: Comparison of different training schemes on Qwen2.5. The superscript in "LoRA" (e.g., ^{4}, ^{8}, etc.) indicates the rank value used for each variant.

memory usage and inference speed.

Given the strong performance of R-LoRA+, we further ask:

RQ 2: *What explains the effective multi-task generalization of R-LoRA+ in the presence of high head matrix similarity, and what does this reveal about the principles of multi-task generalization in LoRA?*

3.3 Task-Shared vs. Task-Specific Learning

We discuss why R-LoRA+ achieves superior performance despite its high head matrix similarity. We hypothesize that multi-task learning may benefit from two complementary directions: (1) enhancing task-specific knowledge discrimination and specialization, and (2) focusing on shared knowledge across tasks. In R-LoRA, the head matrices are initialized with non-zero values and exhibit large gradient norms during early training, enabling rapid capture of task-related knowledge. The dynamic routing mechanism further encourages each head to specialize in distinct knowledge, promoting task-specific learning. In contrast, R-LoRA+ removes the Router and simply averages the outputs of all heads during training. This forces the head matrices to converge toward shared representations, emphasizing the acquisition of cross-task generalizable features. The superior performance of R-LoRA+ suggests that multi-task generalization may rely more heavily on learning shared knowledge across tasks than on enforcing task-specific specialization.

This observation builds upon findings from previous studies on HydraLoRA(Tian et al., 2024) and R-LoRA(Liu et al., 2025), which show that the down-projection matrix $\mathbf{A}$ in LoRA primarily captures cross-task generalizable knowledge, while the head matrix $\mathbf{B}$ tends to capture task-specific features. Inspired by this, we pose a new question:

RQ 3: *Can increasing the rank of LoRA enhance the expressive capacity of matrix* $\mathbf{A}$*, thereby improving multi-task generalization by better capturing shared knowledge?*

4 Increasing the Rank is All You Need

To evaluate the multi-task generalization ability of the models, we adopt the experimental setup from HydraLoRA and fine-tune models on a diverse subset of the Flanv2 dataset, which includes tasks spanning commonsense reasoning, language understanding, question answering, and so on. Llama2 and Qwen2.5 are used as base models to ensure compatibility and comparability across architectures. The fine-tuned models are evaluated on the Big-Bench Hard (BBH) benchmark, a challenging suite of tasks designed to assess reasoning capabilities in language models. BBH covers a wide range of domains such as logical reasoning, symbolic manipulation, algorithmic tasks, and multi-step question answering, all of which require strong generalization beyond memorization. More details about the dataset composition are provided in Appendix B.3.

Our key finding is that, surprisingly, simply increasing the rank of standard LoRA can achieve multi-task generalization performance on par with more sophisticated multi-task variants such as LoRA MoE and HydraLoRA, without requiring complex architectural modifications. This is demonstrated in Table 2, where we

observe that when the rank of standard LoRA is scaled to match the parameter budget of these variants, its performance becomes highly competitive. Furthermore, as shown in Table 3, on the more recent Qwen2.5 model, the multi-task generalization ability of LoRA improves steadily with increasing rank. Across various experimental settings, LoRA achieves performance comparable to HydraLoRA and R-LoRA, reinforcing the conclusion that enhancing the rank alone suffices for strong multi-task adaptation.

This result highlights that Multi-Adapter and Multi-Head structure may not be essential for effective multi-task learning, and that a simple, higher-rank LoRA module can achieve competitive performance while offering better deployment efficiency due to its mergeable weights.

5 Extended Method

Our analysis shows that high similarity among head matrices does not necessarily harm multi-task performance. In particular, R-LoRA+, a simplified variant of multi-head LoRA without dynamic routing, achieves strong generalization despite having highly similar heads. This suggests that focusing on shared, transferable knowledge across tasks may be more important than enforcing task-specific specialization.

Following this insight, and consistent with findings in HydraLoRA (Tian et al., 2024) and R-LoRA (Liu et al., 2025), we confirm that the down-projection matrix $\mathbf{A}$ primarily captures task-shared features, while the head matrices B_i encode task-specific knowledge. Notably, simply increasing the rank of standard LoRA can match the performance of complex multi-head variants, indicating that structural complexity is not essential when model capacity is appropriately scaled.

5.1 Align-LoRA

To further enhance multi-task generalization, we propose to explicitly encourage the model to learn task-invariant representations. To this end, we incorporate the Maximum Mean Discrepancy (MMD) framework (Sejdinovic et al., 2013) into LoRA-based parameter-efficient fine-tuning, with a particular focus on its multi-kernel extension, MK-MMD (Gretton et al., 2012). To the best of our knowledge, this work is the first to apply MMD in multi-task LoRA adaptation. While MMD has been widely used in domain adaptation and representation learning (Pan et al., 2010), its potential for aligning task-specific features in multi-task settings remains underexplored.

Unlike traditional applications that focus on input or hidden-layer alignment, we apply MK-MMD directly to the output representations of the LoRA down-projection matrix $\mathbf{A}$, promoting shared knowledge across tasks while retaining task-specific expressiveness. Let $\mathcal{T} = \{T_1, T_2, \ldots, T_M\}$ denote a set of M tasks, each associated with its own input distribution p_{T_i}. The output of the LoRA down-projection matrix for task T_i is defined as:

$$\phi_{T_i}(\mathbf{x}) = A \cdot X_{T_i}, \tag{8}$$

where X_{T_i} represents the contextualized input embedding for task T_i.

To encourage cross-task generalization, we minimize the MK-MMD loss between all task pairs (T_i, T_j), formulated as:

$$\begin{aligned}\mathcal{L}_{\text{MK-MMD}} = \sum_{i=1}^{M} \sum_{j=i+1}^{M} \sum_{k \in \mathcal{K}} \\ \Big\| \mathbb{E}_{\mathbf{x} \sim p_{T_i}}[\phi_{T_i}(\mathbf{x})] \\ - \mathbb{E}_{\mathbf{y} \sim p_{T_j}}[\phi_{T_j}(\mathbf{y})] \Big\|^2_{\mathcal{H}_k}.\end{aligned} \tag{9}$$

This loss forces the LoRA module to learn task-invariant features by reducing distributional shifts across tasks in the RKHS space. The adaptive kernel selection mechanism of MK-MMD ensures that the model retains task-specific expressiveness while prioritizing shared knowledge.

In the context of LLM fine-tuning, we incorporate this loss as a regularization term into the standard language modeling objective. Specifically, the total loss function is defined as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{lm}} + \lambda \cdot \mathcal{L}_{\text{MK-MMD}}, \tag{10}$$

where $\mathcal{L}_{\text{lm}}$ denotes the language modeling (or sequence-to-sequence) loss for the current task, and λ controls the influence of the MK-MMD regularization.

We denote this approach as **Align-LoRA**, which introduces a novel direction for improving multi-task generalization within the framework of LoRA. By explicitly aligning task-shared representations through the MK-MMD loss, Align-LoRA encourages models to learn shared knowledge across tasks, enhancing their ability to generalize beyond individual task-specific patterns.

Metrics	LoRA^8	LoRA^{10}	HydraLoRA	R-LoRA	Align-LoRA^8	Align-LoRA^{10}
Qwen2.5-7B	45.61	48.36	47.38	48.32	47.53	**49.24**
Llama3-8B	42.58	44.89	44.03	45.01	45.42	**46.14**
Rank	8	9	10	4	4	4
% Param	0.20	0.25	0.25	0.25	0.20	0.25
A/B	1/1	1/1	1/3	1/3	1/1	1/1

Table 4: Multi-task generalization performance of different LoRA variants on Qwen2.5-7B and LLaMA3-8B, evaluated on Big-Bench Hard (BBH).

A key advantage of Align-LoRA is its compatibility with various LoRA-based adaptation strategies. The representation alignment mechanism can be seamlessly integrated into different initialization schemes. Importantly, unlike Multi-Adapter/Multi-Head LoRA variants that rely on dynamic routing mechanisms during inference, Align-LoRA does not introduce any additional modules that would increase computational or memory overhead. As a result, the trained adapter weights in Align-LoRA can be merged into the base model's parameters at deployment time, eliminating the need for separate adapter computation during inference. This property ensures both efficiency and practicality, making Align-LoRA a lightweight yet effective solution for multi-task adaptation.

5.2 Experiment

In this section, we evaluate the performance of Align-LoRA. In this section, we evaluate the performance of **Align-LoRA** in comparison to standard LoRA and its multi-head variants. To provide a comprehensive assessment of multi-task generalization capabilities. For detailed dataset information, please refer to the Appendix B.4. For evaluation, we use the Big-Bench Hard (BBH) benchmark, which measures the model's ability to generalize across complex reasoning tasks rather than simply memorizing answers. This setup enables us to assess cross-task generalization.

As shown in Table 4, we evaluate the multi-task generalization performance of Align-LoRA on two recent large language models, Qwen2.5-7B and LLaMA3-8B, under various LoRA configurations. Despite variations in the training data, the results consistently demonstrate that increasing the rank leads to improved performance across tasks. Notably, Align-LoRA further strengthens this trend by explicitly aligning task-specific representations through MK-MMD, thereby promoting the learning of shared, task-agnostic knowledge.

Compared to standard LoRA and multi-head variants with comparable parameter budgets, Align-LoRA achieves superior performance on BBH without introducing any additional trainable parameters. This demonstrates the effectiveness of representation-level alignment as a means to improve multi-task generalization within the parameter-efficient fine-tuning framework. *Our work provides concrete evidence that task-shared knowledge alignment is a viable direction for efficient multi-task generalization.*

6 Conclusion

In this work, we investigate the multi-task generalization capabilities of LoRA and propose a simplified variant, **R-LoRA+**. Our analysis reveals that head matrices in multi-head structures often exhibit high similarity, suggesting that structural complexity may not be essential for effective multi-task learning.

Our analysis reveals that simply increasing the rank of LoRA achieves comparable performance to multi-head variants, suggesting that complex architectural designs may be unnecessary for multi-task generalization. Building on this insight, we propose **Align-LoRA**, a lightweight yet versatile method that enhances generalization through MK-MMD-based alignment of task representations. Our approach is compatible with different initialization strategies, introduces no extra trainable parameters, and maintains LoRA's mergeable property while being more efficient for practical deployment than multi-head alternatives.

Our work demonstrates that capturing shared knowledge across tasks is more crucial for multi-task generalization than pursuing structural diversity. We further validate that representation alignment provides an effective pathway to enhance such generalization capability.

7 Limitation

Despite the promising results of R-LoRA+ and Align-LoRA, several limitations should be acknowledged. 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. 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.

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.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2006. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19.

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, Yi Chang, and Yuan Wu. 2024a. Balora: Bias-alleviating low-rank adaptation to mitigate catastrophic inheritance in large language models. *arXiv preprint arXiv:2408.04556*.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024b. 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).

Arthur Gretton, Dino Sejdinovic, Heiko Strathmann, Sivaraman Balakrishnan, Massimiliano Pontil, Kenji Fukumizu, and Bharath K Sriperumbudur. 2012. Optimal kernel choice for large-scale two-sample tests. *Advances in neural information processing systems*, 25.

Chenlu Guo, Yuan Wu, and Yi Chang. 2025. Nlora: Nyström-initiated low-rank adaptation for large language models. *Preprint*, arXiv:2502.14482.

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*.

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*.

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.

Jinda Liu, Yi Chang, and Yuan Wu. 2025. R-lora: Random initialization of multi-head lora for multi-task learning. *Preprint*, arXiv:2502.15455.

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.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. 2015. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR.

Yuren Mao, Yuhang Ge, Yijiang Fan, Wenyi Xu, Yu Mi, Zhonghao Hu, and Yunjun Gao. 2025. A survey on lora of large language models. *Frontiers of Computer Science*, 19(7):197605.

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.

Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. 2010. Domain adaptation via transfer component analysis. *IEEE transactions on neural networks*, 22(2):199–210.

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*.

Dino Sejdinovic, Bharath Sriperumbudur, Arthur Gretton, and Kenji Fukumizu. 2013. Equivalence of distance-based and rkhs-based statistics in hypothesis testing. *The annals of statistics*, pages 2263–2291.

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.

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 Head matrices analysis of R-LoRA

The analysis of head matrices in R-LoRA is presented in Figure 4

A.2 T-SNE analysis

The T-SNE analysis of head matrices in HydraLoRA is shown in Figure 5.

B Datasets

B.1 Head Matrices in Multi-Head LoRA

In the section 3.1, 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 Multi-Task Performance Comparison

In the section 3.2, We fine-tune Qwen2.5-3B on five tasks:

1. **Natural Language Inference**: QNLI (Wang, 2018)
2. **Physical Question Answering**: PiQA (Bisk et al., 2020)
3. **Word Relation Reasoning**: Winogrande
4. **Closed-Book Question Answering**: ARC
5. **Mathematical Reasoning**: GSM8K

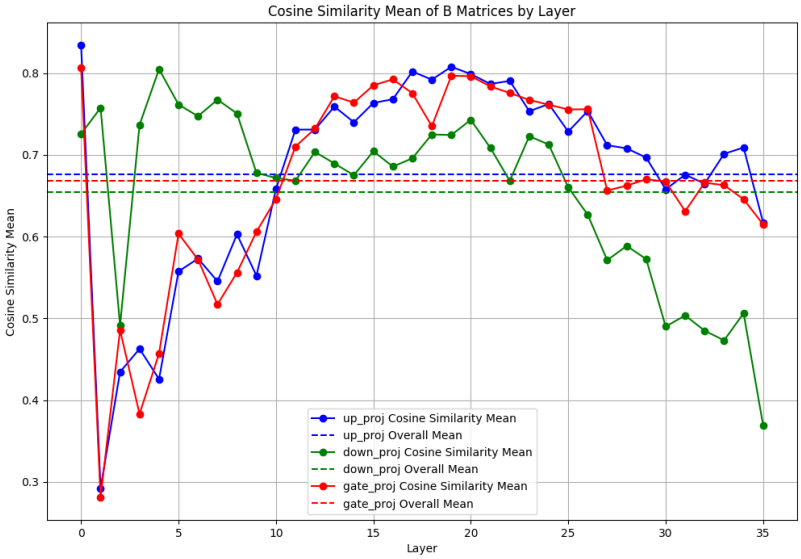

Figure 4: Cosine similarity among head matrices in R-LoRA. "Overall mean" represents the average similarity across all layers.

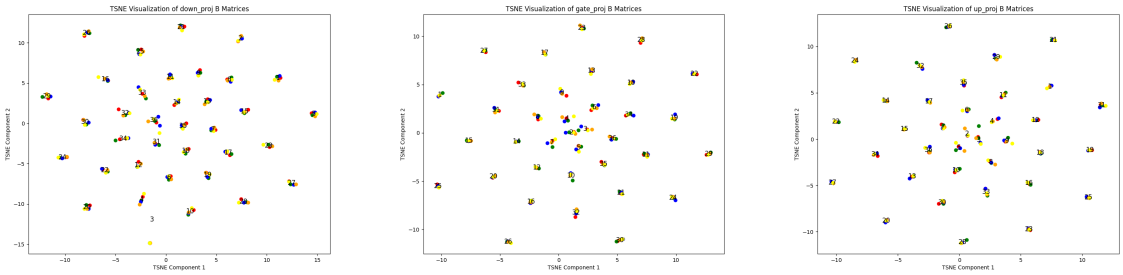

Figure 5: T-SNE analysis of head matrices in HydraLoRA

B.3 Increasing the Rank is All You Need

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 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 Experiment

In the section 5.2, We fine-tune Qwen2.5-7B and Llama3-8B on five tasks. Then we evaluate it with the Big-Bench Hard (BBH) benchmark. We summarize the details of the used datasets as follows:

1. **Natural Language Inference**: QNLI (Wang, 2018)

2. **Physical Question Answering**: PiQA

3. **Word Relation Reasoning**: Winogrande

4. **Closed-Book Question Answering**: ARC

5. **Mathematical Reasoning**: GSM8K

C Implementation Details

The hyperparameters used for training are as follows: a learning rate of 0.0002, `lora_alpha=32`, and trainable LoRA components limited to `q_proj` and `v_proj`. Other modules remain unchanged, following the standard LoRA setup. A dropout rate of 0.2 was applied to the LoRA layers, with a warmup ratio of 0.03. The λ in Align-LoRA is configured within the range of 0.01 to 0.15. Mixed-precision training was enabled using bfloat16, and the learning rate scheduler was set to cosine annealing. The model was trained on NVIDIA 4090 GPU.

D Related Work

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:

$$s_i = R(x)_i = \text{softmax}(Top(W_R^T x, K)) \tag{11}$$

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

$$y = \sum_{i=1}^{n} s_i \cdot E_i(x) \quad \text{(MoE)} \tag{12}$$

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 $\mathbf{A}$ and multiple matrices $B_1, \ldots, B_n$. The shared matrix $\mathbf{A}$ 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 $\mathbf{A}$ for common transformations and the individual matrices B_i for task-specific adjustments.