MIXLORA: ENHANCING LARGE LANGUAGE MODELS FINE-TUNING WITH LORA BASED MIXTURE OF EXPERTS

Dengchun Li*, Yingzi Ma*, Naizheng Wang*, Zhiyuan Cheng[1], Lei Duan*, Jie Zuo*, and Mingjie Tang*

*Sichuan University, Chengdu, China
[1]Purdue University, West Lafayette, USA
mikecovlee@163.com, g19myz@gmail.com, pherenice1125@gmail.com
cheng443@purdue.edu, leiduan@scu.edu.cn, zuojie@scu.edu.cn, tangrock@gmail.com

ABSTRACT

Large Language Models (LLMs) have showcased exceptional performance across a wide array of Natural Language Processing (NLP) tasks. Fine-tuning techniques are commonly utilized to tailor pre-trained models to specific applications. While methods like LoRA have effectively tackled GPU memory constraints during fine-tuning, their applicability is often restricted to limited performance, especially on multi-task. On the other hand, Mix-of-Expert (MoE) models, such as Mixtral 8x7B, demonstrate remarkable performance across multiple NLP tasks while maintaining a reduced parameter count. However, the resource requirements of these MoEs still challenging, particularly for consumer-grade GPUs only have limited VRAM. To address these challenge, we propose MixLoRA, an innovative approach aimed at constructing a resource-efficient sparse MoE model based on LoRA. MixLoRA inserts multiple LoRA-based experts within the feed-forward network block of a frozen pre-trained dense model through fine-tuning, employing a commonly used top-k router. Unlike other LoRA based MoE methods, MixLoRA enhances model performance by utilizing independently configurable attention-layer LoRA adapters, supporting the use of LoRA and its variants for the construction of experts, and applying auxiliary load balance loss to address the imbalance problem of the router. In experiments, MixLoRA achieves commendable performance across all evaluation metrics in both single-task and multi-task learning scenarios. Implemented within the m-LoRA framework, MixLoRA enables parallel fine-tuning of multiple mixture-of-experts models on a single 24GB consumer-grade GPU without quantization, thereby reducing GPU memory consumption by 41% and latency during the training process by 17%.

Keywords Large Language Model · Parameter-efficient Fine-Tuning · MoE · LoRA

1 Introduction

Recently, instruction fine-tuning of large language models (LLMs) [1, 2, 3, 4, 5] for various downstream tasks has achieved impressive proficiency in Natural Language Processing (NLP) [6, 7, 8]. As the scale of parameters increases, LLMs have been demonstrated to be able to identify complex linguistic patterns, thereby enabling the emergence of powerful cross-task generalization capabilities [9]. The paradigm of instruction tuning leads to a trade-off between the computational resources required and the performance achieved on downstream tasks, which has been a valuable facet.

To substantially reduce the computational and memory resources required by traditional fine-tuning processes, Parameter-Efficient Fine-Tuning (PEFT) methodologies have emerged [18, 19, 20, 21, 22, 23]. Among them, Low-Rank Adaptation (LoRA) [23], a popular PEFT method, provides comparable performance to complete fine-tuning on various downstream tasks, and it requires less computational expense. This is accomplished by incorporating adaptation parameters to simulate domain-specific gradient updates while freezing pretrained model weights. However, as introduced in the previous works [17, 24], these LoRA-based methods struggle to handle different tasks simultaneously due to the small number of parameters and catastrophic forgetting, thereby reducing the cross-task generalization performance of LLMs across other diverse tasks. Consequently, some work improves LLM's cross-task generalization ability by

Table 1: Comparison of Similar LoRA MoE Methods. † Denotes MoE Applied to Transformer Block.

Method	Sparse	Router	Attention	Feed Forward	Load Balance
MixLoRA	Yes	Top-k	LoRA	LoRA MoE	Yes
MoRAL [10]	Yes	Top-k	Freeze	LoRA MoE	No
PESC [11]	Yes	Top-2	Freeze	Adapter MoE	Yes
LoRAMoE [12]	No	Fixed	Freeze	LoRA MoE	Yes
MOLA [13]	Yes	Top-2	LoRA MoE†		No
MoELoRA [14]	Yes	Top-k	LoRA MoE†		Yes
MoCLE [15]	Yes	Top-1 with a Fixed	LoRA MoE†		No
MOELoRA [16]	Yes	Top-1 Model Level	LoRA MoE†		No
MoA [17]	Yes	Top-1 Sequence Level	LoRA MoE	LoRA MoE	No

introducing different LoRA modules or experts to process tokens from different tasks. For example, at the task level, AdapterFusion [25] and LoRAHub [24] integrate task-specific knowledge into various pre-trained domain adapters by designing an extra attention fusion module and an element-wise LoRA composition, respectively. At the token level, PESC [26] presented a method for transforming dense models into sparsely activated mixture-of-experts (MoE) models during the instruction tuning stage, with each expert controlled by a unique gating network, activated based on the distinct nature of the input data from diverse tasks.

Naturally, some recent work has attempted to combine LoRA with MoE, constructing the efficient sparse MoE structure by using multiple LoRA modules to handle different tokens and tasks, as depicted in Table 1. This aims to reduce the number of parameters for fine-tuning while enhancing the LLM's cross-task generalization ability as much as possible. Compared to vanilla LoRA, existing LoRA-MoE methods typically involve adding multiple LoRA modules into different sublayers of the transformer block to form multiple experts (**I**) [10, 14, 17] and employing a router to assign experts to handle different tokens dynamically (**II**) [10, 11, 12, 15, 16, 17]. To avoid the uneven distribution of tokens among experts, some methods also consider incorporating a load balance loss to mitigate this issue (**III**) [10, 11]. These three points are considered the main distinctions in existing LoRA-MoE methods. We observe these methods that only construct MoE on the FFN layers achieve performance comparable to or surpass those that construct MoE across the entire transformer block. An important reason is that the FFN encapsulates the transformer block's knowledge effectively [27]. Furthermore, studies such as ST-MoE [28] suggest that fine-tuning the attention layer can bring significant benefits to MoE models.

In this paper, we propose **MixLoRA**, a PEFT method that constructs multiple LoRA experts based on the frozen shared FFN block during the fine-tuning stage. This approach facilitates the construction of an efficient sparse MoE model, enabling improved performance across multiple downstream tasks. Unlike other LoRA-MoE methods, **MixLoRA** enhances model performance by utilizing independently configurable attention-layer LoRA adapters, supporting LoRA and its variants for the construction of experts. We also apply auxiliary load balance loss to address the imbalance problem of the router. In addition, we find the prior LoRA-MoE methods fail to consider the training and inference performance, which is a non-trivial task requiring efforts from both engineering and algorithmic aspects. Existing training frameworks, such as Megatron-LLM [29] and DeepSpeed [30], mainly focus on optimizing training for multiple GPUs and nodes. From the previous work [31], we can see that optimizing the computational overhead of MoE structure and supporting parallel, high-throughput training and inference is not a trivial task. For enhancing efficiency, we leverage the m-LoRA framework [32] to achieve high-throughput parallel training multiple MixLoRA models on a single 24GB consumer-grade GPU [1].

We validate the effectiveness and efficiency of **MixLoRA** across a wide variety of tasks. The quantitative results show that **MixLoRA** outperforms the existing state-of-the-art PEFT method DoRA [33] method in both single-task and multi-task learning scenarios while offering improved accuracy with modest increases in memory use and latency. For single-task learning, it achieves an average accuracy improvement of **7.7%** on LLaMA-2 7B than vanilla LoRA. Furthermore, **MixLoRA** also significantly outperforms DoRA on multi-task learning by **7.2%** accuracy while achieving up to **1.2×** speedup and **1.6×** memory reduction.

In summary, our contributions in this paper are as follows:

1. We introduce MixLoRA, an efficient fine-tuning method that constructs the sparse mixture-of-experts (MoE) model by integrating multiple LoRA experts into the frozen shared feed-forward network (FFN) block of dense

[1] GitHub: https://github.com/mikecovlee/mixlora

models. This approach simplifies the creation of an efficient sparse mixture-of-experts model with limited resource, thereby enhancing performance across various downstream tasks.

2. We implement a high throughput structure for the training process. It achieves a 42% reduction in GPU memory usage and a 17% decrease in latency per token during training on a single 24GB consumer-grade GPU. Furthermore, by optimizing the additional overhead in the computation process, we achieved approximately a 10% increase in the speed of forward propagation for MixLoRA compared to running it without optimization.
3. We design and conduct a series of experiments, providing comprehensive evaluations on several commonly used benchmarks such as ARC [34], BoolQ [35], OpenBookQA [36], and PIQA [37]. The results demonstrate that MixLoRA exhibits superior performance in handling various downstream tasks compared to existing fine-tuning methods. Specifically, it achieves an accuracy higher than LoRA with LLaMA-7B by 7.6% on single-task evaluation and 8% on multi-task evaluation on average, while maintaining computational resource consumption comparable to it.

2 Related Works

2.1 Large Language Models

Recently, Large Language Models (LLMs)[1, 2, 3, 4, 5] have demonstrated remarkable capabilities in natural language processing(NLP) tasks, both in zero-shot and few-shot settings. With the emergence of the Transformer [38] model, which leverages attention mechanisms for enhanced parallelization and efficiency in sequence transduction, LLMs have seen significant improvements in tasks like machine translation, setting new benchmarks for performance and training efficiency. Following this advancement, instruction finetuning [39, 40, 8] has further enabled LLMs to understand human intentions and follow instructions, serving as the foundation of chat systems [41, 42]. However, as the size of LLMs scales up, finetuning them becomes a process that is highly computationally expensive, time-consuming, and memory-intensive. Therefore, to mitigate this issue, various studies explore different approaches: parameter-efficient finetuning (PEFT)[43], distillation[44, 45], quantization [46, 47], pruning [48, 49], etc. In this paper, we primarily focus on the PEFT-based method.

2.2 PEFT

Significant progress has been made in parameter-efficient fine-tuning (PEFT) for LLMs, allowing for significant memory savings by fine-tuning LLMs using adapters. Adapters fall into four main categories: (1) Adapter-based methods involve integrating supplementary trainable modules into the original fixed input. This includes method proposed by BERT [50], which introduce a small, trainable MLP after attention and feed-forward modules. (2) Prompt tuning involves adding extra soft tokens (prompts) to the initial input structure. The primary focus is on exclusively fine-tuning these trainable vectors. Methods such as [20, 51, 52] modify prompts to improve understanding with minimal parameter updates. Additionally, notable methods such as BitFit [21] and IA3 [22] are made available. For a unified view of these adapters, one is directed to the MAM Adapter [53]. (3) Prefix Tuning [19] directs the model to perform specific tasks by adding a specific prefix to the input data, thus adjusting model behavior. (4) LoRA [23, 54] and its derivatives leverage low-rank matrices to decompose linear layer weights, enhancing model performance without introducing any additional computational overhead during inference. For instance,Combining Modular Skills [55] dynamically selects LoRA combinations from an inventory using a routing mechanism tailored to different tasks. VeRA [56] incorporate learnable scaling vectors to adjust shared pairs of frozen random matrices across layers. Furthermore, FedPara [57] concentrates on low-rank Hadamard products for federated learning scenarios. AdaLoRA [58] employ Singular Value Decomposition (SVD) to decompose matrices and prune less significant singular values for streamlined updates. [59] implement weight tying to further reduce the number of trainable parameters. DoRA [33] decomposes pre-trained weights into two components, magnitude and direction, and utilizes LoRA for directional updates during fine-tuning, thereby efficiently reducing the number of trainable parameters.

2.3 Mixture-of-Experts (MoE)

The concept of Mixture-of-Experts (MoE) [60] dates back to as early as 1991, introducing a novel supervised learning approach involving multiple networks (experts), each specialized in handling a subset of training examples [61]. This architecture aimed to mitigate interference effects during training by promoting competition among experts through a gating network, showcasing its efficacy on a multispeaker vowel recognition problem. Advancing from this foundational principle, the MoE model architecture has significantly evolved, particularly within the realm of Large Language Models (LLMs). As LLMs scale by increasing model depth, training challenges and resource demands escalate. The modern incarnation of MoE modifies the traditional feed-forward sub-layer within transformer blocks by incorporating

sparsely activated experts, thereby enabling substantial increases in model width without a corresponding surge in computational demands. Various MoE architectures have emerged, distinguished by their sampling strategies and routing mechanisms. Building on this evolution, LLaVA-MoLE [62] effectively routes tokens to domain-specific experts within Transformer layers, mitigating data conflicts and achieving consistent performance gains over plain LoRA baselines. For other MoE based methods, MoRAL [10]addresses the challenge of adapting large language models (LLMs) to new domains/tasks and enabling them to be efficient lifelong learners. LoRAMoE [12] integrates LoRAs using a router network to alleviate world knowledge forgetting. PESC [11] transitions dense models to sparse models using a Mixture-of-Experts (MoE) architecture, reducing computational costs and GPU memory requirements. MoE-LoRA [14] propose a novel parameter-efficient MoE method with Layer-wise Expert Allocation (MoLA) for Transformer-based models. MoCLE [15] proposes a MoE architecture to activate task-customized model parameters based on instruction clusters As for ours, we integrate LoRA as stochastic experts, reducing computational consumption while expanding model capacity and enhancing LLMs' performance in downstream tasks.

2.4 Fine-tuning Approaches and Frameworks

There are diverse memory and training efficient frameworks designed for fine-tuning LLMs, such as FastChat [8] introduces two benchmarks, MT-bench and Chatbot Arena, to verify the agreement between LLM judges and human preferences. UNIPELT [63] introduces a unified framework to effectively tune pre-trained language models(PLMs) with fewer trainable parameters, especially when training data is limited. Yeh et al. [64] introduce a unified framework within the LoRA family for stable diffusion processes. Notably, our method is based on m-LoRA [32], the first framework for accelerating LoRA fine-tuning, which can dynamically schedule computational and memory resources for multiple LoRA jobs to optimize training throughput and resource utilization while maintaining model performance.

2.5 Dense LLMs and Evaluation Methods

In the realm of foundation language models, LLaMA [65],with models ranging from 7B to 65B parameters, demonstrates that large-scale models can be effectively trained on publicly available datasets, achieving superior performance without proprietary data. Mistral [66] innovates with grouped-query attention and sliding window attention mechanisms, enhancing inference speed and efficiency, while outperforming larger models in reasoning, mathematics, and code generation tasks. About training tasks, ARC [34] challenges systems with science questions requiring deep reasoning, underlining the need for beyond surface-level understanding. BoolQ [35] explores the complexity of natural yes/no questions, demonstrating the importance of inferential reasoning. OpenBookQA [36] assesses understanding by combining scientific facts with common knowledge, emphasizing multi-hop reasoning and the integration of external knowledge. PIQA [37] introduces physical commonsense reasoning, highlighting the gap in current systems' ability to understand and reason about the physical world. Collectively, these tasks encourage the development of models capable of deeper comprehension, inferential reasoning, and integration of diverse knowledge types, addressing critical gaps in natural language understanding and reasoning capabilities.

3 Method

3.1 Overview

The Mixture-of-LoRA-Experts, abbreviated as **MixLoRA**, is a parameter-efficient fine-tuning (PEFT) method used for creating sparse mixture-of-experts models through fine-tuning on dense models such as LLaMA and Mistral. MixLoRA achieves the MoE structure with lower overhead by inserting multiple different LoRA adapters and a top-k router on top of the FFN block of dense models. Furthermore, as analyzed by ST-MoE [28], **MixLoRA** enhances model performance by utilizing independently configurable attention layer LoRA adapters. Building upon this, we further design a resource-efficient framework for **MixLoRA** and other LoRA-MoE methods based on m-LoRA [**?**], which can be employed for parallel training multiple LoRA experts in the MoE layer. Then, by considering the costs of fine-tuning, our framework also supports multiple **MixLoRA** training at the same time, thus reducing 41% of GPU memory consumption and 17% of latency during the training process.

3.2 Architecture

In this section, we introduce **MixLoRA**, a method that leverages LoRA for parameter-efficient fine-tuning (PEFT) to construct sparse mixture-of-experts models through fine-tuning on dense models as Figure 1. **MixLoRA** is constructed from two main parts. The first part involves constructing the sparse MoE block using the vanilla transformer block augmented with LoRAs. The second part utilizes a top-k router to assign each token from various tasks to different

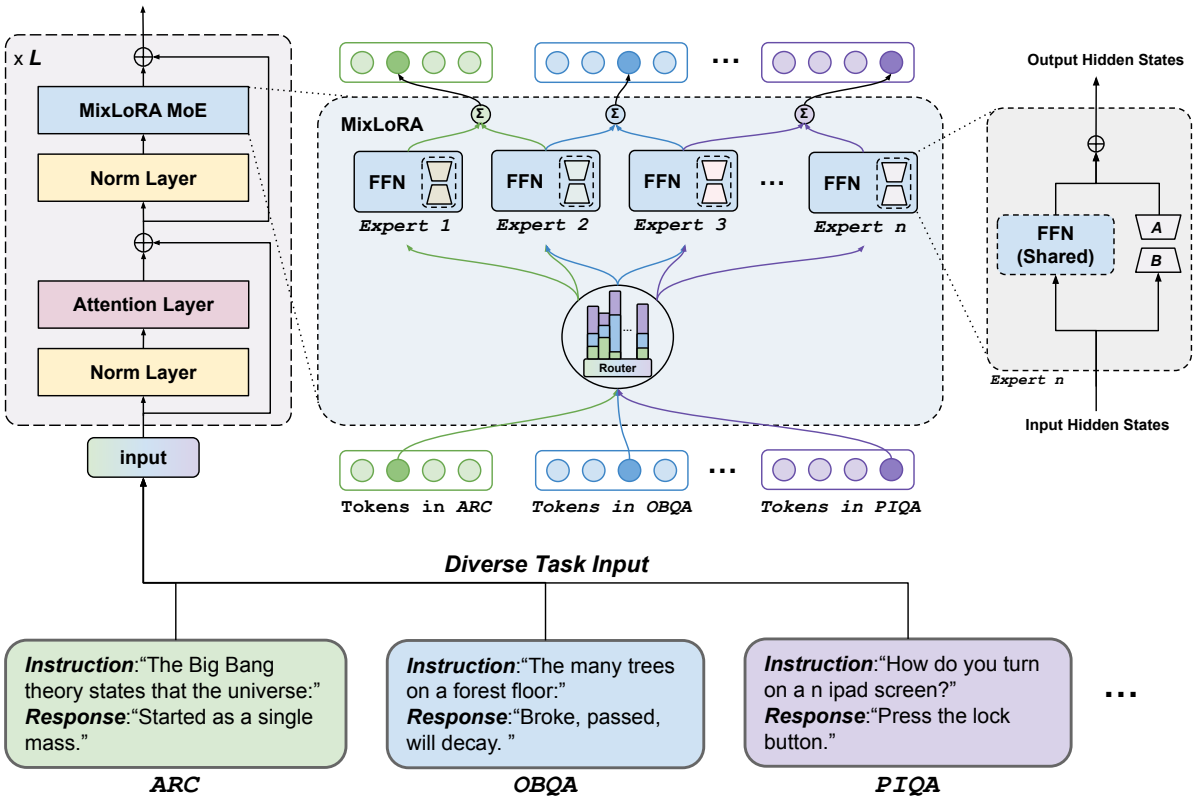


Figure 1: The architecture of MixLoRA transformer block.

expert modules. This approach allows us to harness the power of MoE along with the capabilities of LoRA. Specifically, it only requires loading LoRAs into the GPU memory, thereby avoiding loading inactive FFN (Feed Forward Network) layers into GPU memory, which significantly saves space. Additionally, the FFN layers as experts, equipped with different LoRAs, can process tokens from various tasks, thereby enhancing the multi-task learning capability.

Construct Mix-MoE Block with LoRA based Experts and a Top-k Router. A standard Mixture of Experts (MoE) architecture typically consists of N parallel feed-forward networks (FFNs) as experts, which necessitate a large number of parameters. MixLoRA constructs experts based on the LoRA technique [23], leveraging a shared single feed-forward network (FFN) from the dense model to enhance training and inference efficiency.

Formally, for the traditional transformers architecture, the forward propagation process of the feed-forward network (FFN) block can be simplified as follows:

$$f(x) = x + f_{\text{FFN}}(x) \tag{1}$$

The dense models such as LLaMA and Mistral implement the feed-forward network (FFN) with SwiGLU activation, which comprises three linear components: w_1, w_2, w_3, representing *gate*, *down*, and *up* projection. The forward propagation process of SwiGLU is as follows:

$$f_{\text{FFN}}(x) = w_2(\text{SwiGLU}(w_1(x)) \cdot w_3(x)) \tag{2}$$

The matrix operation of each linear layer with LoRA adapter in this forward propagation can be expressed as:

$$w_i^{'}(x) = W_0 x + \Delta W x = W_0 x + BAx \tag{3}$$

where $W_0 \in \mathbb{R}^{d \times k}$ represents the weights of the frozen feed-forward network of the dense model, and $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$ denote the low-rank decomposition matrices, with the rank $r \ll \min(d, k)$. For MixLoRA, we construct N mix-experts with N pairs of low-rank decomposition matrices B and A, denoted as B_i and A_i, where i indicates the index of the expert. All of the mix-experts share the same frozen W_0 to improve efficiency. The forward process of mix-expert with SwiGLU activation can be expressed as:

$$E(x) = w_2^{'}(\text{SwiGLU}(w_1^{'}(x)) \cdot w_3^{'}(x)) \tag{4}$$

This approach significantly circumvents unnecessary duplication and conserves GPU memory. With these definitions, the forward propagation process of the MixLoRA MoE block can be mathematically expressed as follows:

$$f_{\text{Mix-MoE}}(x) = \sum_{i=1}^{N} G(x)_i E_i(x) \tag{5}$$

In this formulation, $G(\cdot) = \text{Softmax}(\text{TopK}(W_g x))$ represents the top-k router in the MixLoRA MoE block, where the matrix W_g is the trainable parameter matrix of the routing network. Despite an increase in parameters, the experts of the Mix-MoE block are activated sparsely, implying that only a limited subset of experts is used per input token. This approach enhances the capacity of the dense model while maintaining computational efficiency. During inference, the top-k gate router dynamically selects the best k experts for each token. Through this mechanism, the mix-experts and the router work in tandem, enabling the experts to develop varied capacities and efficiently handle diverse types of tasks.

3.3 Training procedure

MixLoRA's training parameters are similar to LoRA, constituting only a small fraction of the dense model. However, the number of parameters during computation is higher than that of the dense model, approaching that of a Mixture of Expert (MoE) model. Taking LLaMA-7B as an example, during the training stage, the parameters of MixLoRA account for approximately 1% to 10%, depending on the configuration, similar to LoRA within the same order of magnitude. However, during inference stage, the number of parameters for MixLoRA varies based on the top-k value. When $K = 2$, the computed parameter count is approximately 13 billion. This is because each expert in MixLoRA shares the same pretrained model's feed-forward network (FFN) layer, and this portion of weights is involved in computations multiple times. However, due to the different LoRA matrices applied to it, the computed results are also different. The number of experts in MixLoRA does not affect the number of parameters activated during inference, but it increases the training parameters and memory overhead, making the model more challenging to converge.

Enhancing Fine-Tuning Capabilities with Attention Layer. MixLoRA further extends its fine-tuning capabilities to encompass the attention layer. Previous studies, such as ST-MoE [28], have suggested that fine-tuning the attention layer can significantly improve performance. To enhance the fine-tuning process with MixLoRA, we integrate LoRA adapters into the attention layer of the dense model. Experimental results demonstrate that the MixLoRA model, fine-tuned with q, k, v, and o projection, consistently achieves superior average scores compared to the identical configuration trained solely with a sparse Mixture-of-Experts (MoE) layer.

Experts Load Balance. Unbalanced load of experts are a big challenge for MoEs, especially for top-k routers, where few experts tend to be more frequently chosen by router. To counter this imbalance problem, we applied load balancing loss to avoid the unbalanced load for each experts when training. Inspired by Switch Transformers, we calculate the auxiliary loss and added it to total loss. Given N experts indexed by $i = 1$ to N and a batch B with T tokens, the auxiliary loss is computed as following:

$$\text{loss} = a \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i \tag{6}$$

where f_i is the fraction of tokens dispatched to expert i,

$$f_i = \frac{1}{T} \sum_{x \in B} \mathbb{1}\{\text{argmax p(x) = i}\} \tag{7}$$

and P_i is the fraction of the router probability allocated for expert i,

$$P_i = \frac{1}{T} \sum_{x \in B} p_i(x). \tag{8}$$

The final loss is multiplied by the expert count N to keep the loss constant as the number of experts varies. Additionally, we utilize $a = 10^{-2}$ as a multiplicative coefficient for auxiliary losses. This coefficient is chosen to strike a balance: it is large enough to ensure load balancing while remaining small enough to not overwhelm the primary cross-entropy objective.

3.4 Discussion of LoRA Variant

Applying LoRA and its Variants with MixLoRA. MixLoRA is also compatible with LoRA variants such as DoRA. DoRA stands for Weight-Decomposed Low-Rank Adaptation, which decomposes the pretrained weights into two

components: magnitude and direction, for fine-tuning. It specifically utilizes LoRA for directional updates to efficiently minimize the number of trainable parameters. DoRA enhances both the learning capacity and training stability of LoRA. With $W_0 \in \mathbb{R}^{d\times k}$ representing the frozen pretrained model weights, $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ denote the low-rank decomposition matrices, where the rank $r \ll \min(d, k)$. The magnitude of DoRA is defined as $m = \|W_0\|_c$, where $\|\cdot\|_c$ represents the vectorwise norm of a matrix across each column. The matrix operation of each linear layer with DoRA adapter in forward propagation of MixLoRA MoE block can be expressed as:

$$w_i^{'}(x) = m\frac{W_0 x + BAx}{\|W_0 + BA\|_c} \tag{9}$$

We denote MixLoRA with DoRA applied as MixDoRA. MixDoRA outperforms MixLoRA in a single-task scenario but lags behind MixLoRA in a multi-task scenario. Experimental results indicate that the extended learning capacity of MixDoRA may require more training and tuning compared to MixLoRA.

3.5 Improve Efficiency via Batch Fusion

A single MixLoRA model based on LLaMA-7B can be easily deployed on a consumer-grade GPU with less than 24GB memory without the need for quantization. In our experiments, when LLaMA-7B was loaded with half-precision, a single MixLoRA model with 8 experts and LoRA rank = 16 only consumed 17.88GB of GPU memory, whereas LoRA with similar training parameters occupied 17.62GB of GPU memory. The additional memory used by MixLoRA primarily comes from activation memory during training, as the number of parameters involved in the computational process of MixLoRA is significantly higher than that of LoRA (approximately 13B).

Given that the pretrained dense model weights in MixLoRA remain frozen, it becomes feasible to maintain two or more MixLoRA models that share the same pretrained dense model weights. Figure 2 illustrates the concept of m-LoRA[32], which leverages Batch Fusion to enhance the training computation efficiency of multiple LoRA models. We have built MixLoRA upon the m-LoRA framework, enabling the fine-tuning of multiple Mixture-of-Experts models on a single 24GB consumer-grade GPU. In experiments, we observed that m-LoRA with MoE optimization can conserve 41% of GPU memory and reduce latency per token by 17% when training two MixLoRA models simultaneously.

Reduce Computational Overhead. We also leverage Batch Fusion further for experts' computation of MixLoRA. From Equation 5, we can observe that the computation process of experts can be parallelized. Since every mix-expert includes a LoRA adapter, the computation process of MixLoRA is more similar to having N LoRA models running simultaneously. This implies that the kernel launch overhead will be more critical compared to the normal multi-LoRA scenario. To mitigate the computational overhead associated with kernel launch, we employ kernel fusion for the mix-experts. Traditionally, we calculate SwiGLU for each expert sequentially. Notably, the calculations of w_1 and w_3 are independent of other variables, allowing parallel processing. Consequently, we optimize computation efficiency by simultaneously sending the data from the output of the normalization layer to both the top-k gate, w_1 and w_3 for computation. Subsequently, we segment the results based on $G(\cdot)$ and forward them to the respective LoRA experts for computation. With $w_i^\dagger, w_3^\dagger$ is the segmented hidden states of w_1, w_3, $\Delta W_i = B_i A_i$ is the low-rank decomposition matrix for each linear, the function os w_1, w_2, w_3 can be replaced with:

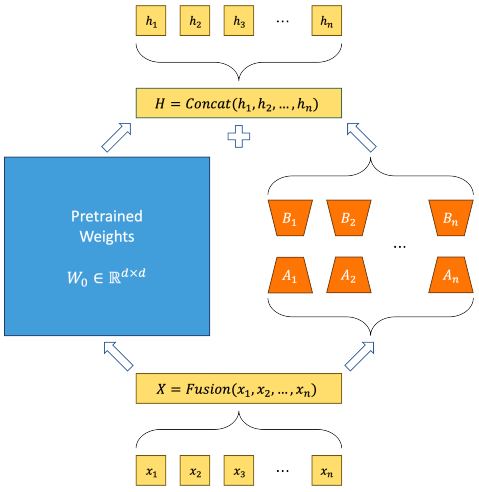


Figure 2: Sharing pre-trained model weights for training multiple LoRA jobs with reduced overhead.

$$\begin{aligned} w_1^\dagger(x) &= \Delta W_1 x + w_1^\dagger \\ w_2^\dagger(x) &= \Delta W_2 x + w_2(x) \\ w_3^\dagger(x) &= \Delta W_3 x + w_3^\dagger \end{aligned}$$

Then, the efficient forward process of mix-experts is as follows:

$$E(x) = w_2^\dagger(\text{SwiGLU}(w_1^\dagger(x))) \cdot w_3^\dagger(x))) \tag{10}$$

This approach significantly enhances computational efficiency, resulting in a training speed improvement of over 10%.

4 Experiments

4.1 Experimental Setup

Dataset. To evaluate the effectiveness of **MixLoRA**, we conduct comprehensive experiments on a variety of supervised fine-tuning datasets in the area of common sense reasoning. ARC [34] evaluates LLM's capabilities using science-based multiple-choice questions. BoolQ [35] features real-world Google queries with binary answers derived from Wikipedia excerpts. OpenBookQA [36] and PIQA [37] present multiple-choice challenges that demand comprehension and logic applied to selected scientific information or physical interactions, respectively. For these datasets, we only use the accuracy metric to quantify their performance.

Table 2: Description of Datasets used in the experiment.

Task	Domain	# Train	# Test	Language	Answer Type
ARC	Natural Science	3,370	3,548	English	Multiple choices
BooLQ	Wikipedia	9,400	3,270	English	Yes/No
OpenBookQA	Science Facts	5,000	500	English	Multiple choices
PIQA	Physical Commonsense	16,000	600	English	Options

Implementation Details. **MixLoRA** is configured with $r = 16$, featuring 8 experts and a top-2 router. We activate q, k, v, o on attention layers, as well as w_1, w_2, w_3 on feed-forward layers for experts. Additionally, to explore **MixLoRA**'s performance with different variants of LoRA, we also conduct **MixLoRA** with DoRA applied, denoted as MixDoRA. The experiments are conducted using 7B pre-trained models on NVIDIA GPUs with 24GB memory, including RTX 3090, RTX A6000, and RTX 4090, while 13B pre-trained models are utilized on NVIDIA RTX A6000. Python 3.10 with m-LoRA version 0.2 and Ubuntu 22.04 are employed for all experiments.

Baselines We utilize commonly used LLaMA-2-7B, LLaMA-2-13B, and the new Mistral-7B for LoRA, DoRA, MixLoRA and MixDoRA, respectively. To ensure the consistency of the parameter sizes across all PEFT methods, we establish LoRA and DoRA with $rank = 80$ and enable q, k, v, o on attention layers, as well as w_1, w_2, w_3 on feed-forward layers as our baseline methods.

4.2 Main Results

Table 3: Accuracy comparison of LLaMA-2 7B/13B and Mistral 7B on commonsense reasoning tasks.

Model	PEFT Method	# Params	ARC-e	ARC-c	BoolQ	OBQA	PIQA	AVG.
LLaMA-2-7B	LoRA	2.6%	73.8	50.9	62.2	80.4	69.9	67.4
	DoRA	2.6%	76.5	**59.8**	71.7	80.6	78.8	73.5
	MixLoRA	2.6%	76.4	58.1	73.8	**84.4**	82.6	75.1
	MixDoRA	2.6%	**78.3**	59.6	**74.2**	**84.4**	**83.6**	**76.0**
LLaMA-2-13B	LoRA	2.2%	83.2	67.6	75.4	83.2	86.7	79.2
	DoRA	2.2%	83.1	67.7	75.1	84.5	**87.8**	79.6
	MixLoRA	2.2%	**83.5**	**69.9**	**77.1**	83.0	86.8	**80.1**
	MixDoRA	2.2%	**83.5**	68.9	75.2	**86.2**	86.0	80.0
Mistral 7B	LoRA	2.7%	85.3	72.5	69.8	87.6	82.1	79.5
	DoRA	2.7%	85.4	72.6	70.4	89.6	89.0	81.4
	MixLoRA	3.1%	**89.4**	74.7	76.1	89.6	**90.1**	**84.0**
	MixDoRA	3.1%	86.8	**76.0**	**76.2**	**89.8**	90.0	83.8

Single-task setup. Table 3 presents the performance of MixLoRA and compares these results with outcomes obtained by employing LoRA and DoRA for fine-tuning. The results demonstrate that the language model with MixLoRA achieves commendable performance across all evaluation methods. Particularly noteworthy is that while DoRA

Table 4: Experimental results of LLaMA-2 7B for multi-task learning on commonsense reasoning tasks. Single-Task LoRA/DoRA (**ST**) is the training of independently trained PEFT modules for each task while Multi-Task LoRA/DoRA (**MT**) shows results of jointly trained PEFT modules using the default **ST** settings. Reported results are accuracy scores.

PEFT Method	# Params (%)	ST/MT	ARC-e	ARC-c	BoolQ	OBQA	PIQA	AVG.
LoRA	2.59	ST	73.8	50.9	62.2	80.4	69.9	67.4
	2.59	MT	61.3	55.7	66.7	71.6	72.4	65.5
			-12.5	4.8	4.5	-8.8	2.5	-1.9
DoRA	2.59	ST	76.5	59.8	71.7	78.6	78.8	73.1
	2.59	MT	64.5	54.1	65.4	75.8	71.9	66.3
			-12.0	-5.7	-6.3	**-2.8**	-6.9	-6.7
MixLoRA	2.64	ST	76.4	58.1	73.8	84.4	82.6	75.1
	2.64	MT	**76.6**	**64.2**	**71.2**	72.8	**82.7**	**73.5**
			0.2	6.1	**-2.6**	-11.6	**0.1**	**-1.5**
MixDoRA	2.64	ST	78.3	59.6	74.2	84.4	83.6	76.0
	2.64	MT	73.3	62.7	68.5	**77.6**	76.8	71.8
			-5.0	**3.1**	-5.7	-6.8	-6.8	-4.2

outperforms LoRA in most evaluations, MixDoRA does not consistently exhibit superior performance compared to MixLoRA, especially with larger models (e.g., LLaMA 13B).

In most evaluations, it can be observed that the fine-tuning results of MixLoRA on LLaMA 7B are very close to the results of LoRA on LLaMA 13B, with an average score difference of only 4.4 points, mainly differing in the ARC test. MixLoRA achieves greater gains in fine-tuning results on Mistral 7B compared to LoRA, surpassing all results obtained on LLaMA 13B. This indicates that MixLoRA effectively extends the model's capacity by building multiple experts on the FFN of the dense model.

Multi-task setup. Table 4 presents the results of MixLoRA in multi-task learning. In contrast to the single-task learning in Table 3, during multi-task learning, we mix training data from ARC, BoolQ, OBQA, and PIQA together to train the model, followed by separate evaluation to investigate the model's generalization ability. The results indicate that compared to single-task learning, all models exhibit some degree of performance degradation in multi-task learning. However, MixLoRA maintains its superiority, with not only a smaller decrease in average score compared to LoRA ($1.5 < 1.9$) but also more balanced scores across different tasks. This suggests that MixLoRA demonstrates stronger generalization ability and mitigates the issue of knowledge forgetting in multi-task learning relative to LoRA.

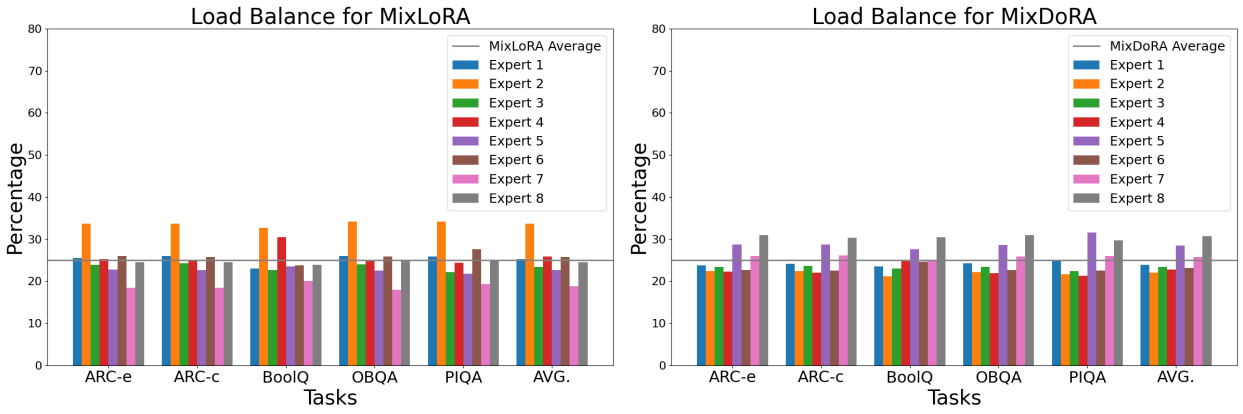
4.3 Analysis of Load Balance

To confirm the effectiveness of MixLoRA in specializing the experts with different tasks, we present the distribution of experts across diverse tasks for MixLoRA and MixDoRA in Figure 3, respectively. Intuitively, we can observe that the expert loads in all tasks are balanced, which means these experts are allocated to different tasks relatively evenly. MixLoRA and MixDoRA exhibit differentiated expert loadings within a multi-task learning framework. Building upon the previous analysis, we contend that the balanced workload in MixLoRA and MixDoRA, indicated by the lower average standard deviation (0.0223 and 0.0328), suggests a homogenous distribution of tasks, which leads to an efficient utilization of expert capabilities within the model.

Furthermore, we find that MixLoRA shows a higher variability in expert loads, implying a degree of specialization where certain experts may be tasked more heavily with particular types of problems or tasks. This can be advantageous if those experts are particularly adept at certain tasks, leading to higher efficiency or accuracy in cross-task generalization. MixDoRA, while still relatively balanced, might allow for a more similar distribution of work where expertise can be leveraged for certain tasks, at the risk of less flexibility if the task demands change. It can be proved in Table 4 where MixDoRA cases a higher drop than MixLoRA in multi-task learning experiments.

4.4 Analysis of Efficiency

We compare the efficiency between our methods and LoRA, and DoRA in terms of memory usage, latency, and performance in Table 5. MixLoRA improves performance over the baseline by 11.3% with a modest increase in



(a) (b)

Figure 3: Distribution of expert loadings. The average workload of the 8 experts in MixLoRA (a) and MixDoRA (b) across various tasks during the evaluation process of multi-task learning is depicted in the figure. The average standard deviation of MixLoRA is smaller than that of MixDoRA ($0.0223 < 0.0328$). However, both standard deviations are small enough, indicating that the workload of these experts is relatively balanced.

latency (54.6%) and memory (1.4%). MixDoRA further improves performance by 12.7% but at a high cost in latency (367.1%) and a similar memory increase (1.7%). Benefiting from the m-LoRA framework, we can train multiple MixLoRAs and MixDoRAs at the same time. When scaled (x2), both methods maintain performance gains but with a significant reduction in memory efficiency. MixLoRA exhibits a more balanced trade-off, while MixDoRA offers higher performance at the expense of much greater latency.

Table 5: Trade-off between memory, latency, and performance.

Method	**Score**		**Latency**		**Memory**	
	AVG.	%	MS.	%	GB.	%
LoRA	67.4	100.0%	4.16	100.0%	17.6	100.0%
DoRA	73.1	108.4%	6.91	166.0%	17.6	100.0%
MixLoRA	75.1	111.3%	6.43	154.6%	17.9	101.4%
MixDoRA	76.0	112.7%	19.44	467.1%	17.9	101.7%
MixLoRA $\times 2$	75.1	111.3%	5.72	137.5%	21.3	60.4%
MixDoRA $\times 2$	76.0	112.7%	18.90	454.2%	21.9	62.2%

5 Conclusion

In this paper, we introduced MixLoRA, a parameter-efficient fine-tuning method that constructs multiple LoRA experts and a top-k router on top of the frozen shared feed-forward network block of dense models through fine-tuning. This approach facilitates the construction of an efficient sparse mixture-of-experts model, enabling improved performance across multiple downstream tasks. Building on the m-LoRA framework, we achieved high-throughput parallel training, inference, and evaluation of multiple MixLoRA models on a single 24GB consumer-grade GPU. By optimizing the additional overhead in the computation process, we increased the forward computation speed of MixLoRA by approximately 10% compared to running it without optimization. In experiments, MixLoRA achieves commendable performance across all evaluation metrics in both single-task and multi-task learning scenarios. Implemented within the m-LoRA framework, MixLoRA enables parallel fine-tuning, inference, and evaluation of multiple mixture-of-experts models on a single 24GB consumer-grade GPU without quantization, thereby reducing GPU memory consumption by 41% and latency during the training process by 17%.

References

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. ArXiv, abs/2005.14165, 2020.

[2] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier García, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Díaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. J. Mach. Learn. Res., 24:240:1–240:113, 2022.

[3] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and L. Sifre. Training compute-optimal large language models. ArXiv, abs/2203.15556, 2022.

[4] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. ArXiv, abs/2302.13971, 2023.

[5] Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. ArXiv, abs/2307.09288, 2023.

[6] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. arXiv preprint arXiv:2210.11416, 2022.

[7] Srinivasan Iyer, Xi Victoria Lin, Ramakanth Pasunuru, Todor Mihaylov, Daniel Simig, Ping Yu, Kurt Shuster, Tianlu Wang, Qing Liu, Punit Singh Koura, et al. Opt-iml: Scaling language model instruction meta learning through the lens of generalization. arXiv preprint arXiv:2212.12017, 2022.

[8] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

[9] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. arXiv preprint arXiv:2206.07682, 2022.

[10] Shu Yang, Muhammad Asif Ali, Cheng-Long Wang, Lijie Hu, and Di Wang. Moral: Moe augmented lora for llms' lifelong learning. arXiv preprint arXiv:2402.11260, 2024.

[11] Haoyuan Wu, Haisheng Zheng, and Bei Yu. Parameter-efficient sparsity crafting from dense to mixture-of-experts for instruction tuning on general tasks. arXiv preprint arXiv:2401.02731, 2024.

[12] Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. Loramoe: Alleviate world knowledge forgetting in large language models via moe-style plugin, 2024.

[13] Chongyang Gao, Kezhen Chen, Jinmeng Rao, Baochen Sun, Ruibo Liu, Daiyi Peng, Yawen Zhang, Xiaoyuan Guo, Jie Yang, and VS Subrahmanian. Higher layers need more lora experts, 2024.

[14] Tongxu Luo, Jiahe Lei, Fangyu Lei, Weihao Liu, Shizhu He, Jun Zhao, and Kang Liu. Moelora: Contrastive learning guided mixture of experts on parameter-efficient fine-tuning for large language models, 2024.

[15] Yunhao Gou, Zhili Liu, Kai Chen, Lanqing Hong, Hang Xu, Aoxue Li, Dit-Yan Yeung, James T. Kwok, and Yu Zhang. Mixture of cluster-conditional lora experts for vision-language instruction tuning, 2024.

[16] Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. Moelora: An moe-based parameter efficient fine-tuning method for multi-task medical applications. arXiv preprint arXiv:2310.18339, 2023.

[17] Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Yu Han, and Hao Wang. Mixture-of-loras: An efficient multitask tuning for large language models, 2024.

[18] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In International Conference on Machine Learning, pages 2790–2799. PMLR, 2019.

[19] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), abs/2101.00190, 2021.

[20] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Conference on Empirical Methods in Natural Language Processing, 2021.

[21] Elad Ben-Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. ArXiv, abs/2106.10199, 2021.

[22] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. ArXiv, abs/2205.05638, 2022.

[23] J. Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. ArXiv, abs/2106.09685, 2021.

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

[25] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. arXiv preprint arXiv:2005.00247, 2020.

[26] Haoyuan Wu, Haisheng Zheng, and Bei Yu. Parameter-efficient sparsity crafting from dense to mixture-of-experts for instruction tuning on general tasks, 2024.

[27] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories, 2021.

[28] Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models, 2022.

[29] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020.

[30] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. In ICML 2022, January 2022.

[31] Weilin Cai, Juyong Jiang, Le Qin, Junwei Cui, Sunghun Kim, and Jiayi Huang. Shortcut-connected expert parallelism for accelerating mixture-of-experts. arXiv preprint arXiv:2404.05019, 2024.

[32] Zhengmao Ye, Dengchun Li, Jingqi Tian, Tingfeng Lan, Jie Zuo, Lei Duan, Hui Lu, Yexi Jiang, Jian Sha, Ke Zhang, and Mingjie Tang. Aspen: High-throughput lora fine-tuning of large language models with a single gpu. ArXiv, abs/2312.02515, 2023.

[33] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation, 2024.

[34] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018.

[35] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions, 2019.

[36] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering, 2018.

[37] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019.

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[39] Hyung Won Chung, Le Hou, S. Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Wei Yu, Vincent Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed Huai hsin Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. ArXiv, abs/2210.11416, 2022.

[40] Srinivas Iyer, Xi Victoria Lin, Ramakanth Pasunuru, Todor Mihaylov, Daniel Simig, Ping Yu, Kurt Shuster, Tianlu Wang, Qing Liu, Punit Singh Koura, Xian Li, Brian O'Horo, Gabriel Pereyra, Jeff Wang, Christopher Dewan, Asli Celikyilmaz, Luke Zettlemoyer, and Veselin Stoyanov. Opt-iml: Scaling language model instruction meta learning through the lens of generalization. ArXiv, abs/2212.12017, 2022.

[41] OpenAI. Chatgpt: Optimizing language models for dialogue. https://openai.com/blog/chatgpt, 2022.

[42] OpenAI. Gpt-4 technical report, 2023.

[43] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022.

[44] Zechun Liu, Barlas Oğuz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free quantization aware training for large language models. ArXiv, abs/2305.17888, 2023.

[45] Guangxuan Xiao, Ji Lin, and Song Han. Offsite-tuning: Transfer learning without full model. ArXiv, abs/2302.04870, 2023.

[46] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. ArXiv, abs/2210.17323, 2022.

[47] Guangxuan Xiao, Ji Lin, Mickael Seznec, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. ArXiv, abs/2211.10438, 2022.

[48] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. ArXiv, abs/2301.00774, 2023.

[49] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. ArXiv, abs/2305.11627, 2023.

[50] Asa Cooper Stickland and Iain Murray. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning, 2019.

[51] Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. Ppt: Pre-trained prompt tuning for few-shot learning. ArXiv, abs/2109.04332, 2021.

[52] Zhen Wang, Rameswar Panda, Leonid Karlinsky, Rogério Schmidt Feris, Huan Sun, and Yoon Kim. Multitask prompt tuning enables parameter-efficient transfer learning. ArXiv, abs/2303.02861, 2023.

[53] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In International Conference on Learning Representations, 2022.

[54] Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. Lorahub: Efficient cross-task generalization via dynamic lora composition. ArXiv, abs/2307.13269, 2023.

[55] Edoardo M. Ponti, Alessandro Sordoni, Yoshua Bengio, and Siva Reddy. Combining modular skills in multitask learning, 2022.

[56] Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. Vera: Vector-based random matrix adaptation, 2024.

[57] Nam Hyeon-Woo, Moon Ye-Bin, and Tae-Hyun Oh. Fedpara: Low-rank hadamard product for communication-efficient federated learning, 2023.

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

[59] Adithya Renduchintala, Tugrul Konuk, and Oleksii Kuchaiev. Tied-lora: Enhacing parameter efficiency of lora with weight tying, 2023.

[60] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. Neural Computation, page 79–87, Feb 1991.

[61] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. Neural Computation, 3:79–87, 1991.

[62] Shaoxiang Chen, Zequn Jie, and Lin Ma. Llava-mole: Sparse mixture of lora experts for mitigating data conflicts in instruction finetuning mllms, 2024.

[63] Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Wen tau Yih, and Madian Khabsa. Unipelt: A unified framework for parameter-efficient language model tuning, 2022.

[64] Shin-Ying Yeh, Yu-Guan Hsieh, Zhidong Gao, Bernard BW Yang, Giyeong Oh, and Yanmin Gong. Navigating text-to-image customization: From lycoris fine-tuning to model evaluation. arXiv preprint arXiv:2309.14859, 2023.

[65] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

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