Mixture-of-LoRAs: An Efficient Multitask Tuning for Large Language Models

Wenfeng Feng   Chuzhan Hao   Yuewei Zhang   Yu Han   Hao Wang[*]
Alibaba Cloud, Alibaba Group
{wenfeng.fwf,liyou.zyw,hy226261}@alibaba-inc.com
chuzhanh@gmail.com,cashenry@126.com

Abstract

Instruction Tuning has the potential to stimulate or enhance specific capabilities of large language models (LLMs). However, achieving the right balance of data is crucial to prevent catastrophic forgetting and interference between tasks. To address these limitations and enhance training flexibility, we propose the **M**ixture-**o**f-LoR**A**s (MoA) architecture – a novel and parameter-efficient tuning method designed for multi-task learning with LLMs. In this paper, we start by individually training multiple domain-specific LoRA modules using corresponding supervised corpus data. These LoRA modules can be aligned with the expert design principles seen in Mixture-of-Experts (MoE). Subsequently, we combine the LoRAs using an explicit routing strategy and introduce domain labels to facilitate multi-task learning, which helps prevent interference between tasks and ultimately enhances the performance of each individual task. Furthermore, each LoRA model can be iteratively adapted to new domains, allowing for quick domain-specific adaptation. Experiments on diverse tasks demonstrate superior and robust performance of our approach, which will also further promote the application of domain-specific LLMs.

1 Introduction

Large language models (LLMs) have played a pivotal role in expediting the advancement of natural language processing (NLP), offering a versatile and task-agnostic foundation that underpins an extensive array of applications. The intrinsic diversity found in domain-specific data poses a substantial challenge in training a general-purpose base LLMs. Consequently, there has been a surge in the adoption of domain-specific LLMs to tackle intricate problems within specialized domains, such as SQL-PaLM (Sun et al., 2023),
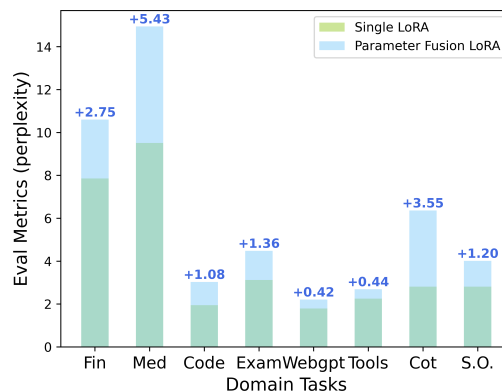

Figure 1: Comparing different LoRA models using multi-task parameter fusion method and corresponding task method.

BloombergGPT (Wu et al., 2023), ChatLaw (Cui et al., 2023), pdfGPT (Tripathi, 2023). In real-world application scenarios, the demand often arises for a multitude of customized capabilities. LLMs with multiple customized capabilities can efficiently address a diverse range of user problems, and each specific functional module can be optimized individually.

Domain specification techniques are key to make large language models disruptive in various applications (Zhao et al., 2023). To learn sufficient domain knowledge and not lose basic capability, adapter-based fine-tuning methods (e.g., Adapters (Houlsby et al., 2019), LoRA (Hu et al., 2021)) introduce a limited number of domain-specific parameters to retain domain-related knowledge and do not need to fine-tuning all parameters of the pre-trained model, which can effectively reduce the training cost of LLMs.

In order to obtain multiple customized capabilities, the simplest and efficient fine-tuning approach is to directly mix data from multiple domains together and only add one LoRA module for instruction fine-tuning. The second is some

---

[*]Corresponding author

two-stage approaches. We first train multiple domain LoRA modules individually, and then introduce a domain classifier to select appropriate LoRA model. In addition, Pfeiffer et al. (2020); Huang et al. (2023) add the *AdapterFusion* layer and *element-wise* LoRA composition to implicitly fuse parameter knowledge of multiple task adapters. The MoE (Shen et al., 2023) introduces multiple experts to process different types of input data. Jiang et al. (2023) is to consider the outputs of ensemble LLMs comprehensively.

However, these approaches have two main issues. First, in the practical application scenario, there are often multiple destructive domain tasks with heterogeneous and imbalance training data, as well as limited computing resources. Therefore, the implicit parameter fusion methods exist mutual disturbance, which results in degraded model performance on in-domain tasks, as illustrated in Figure 1. We often focus more on the domain-specific expertise of a domain-specific LLM than its generalization performance. In addition, the MoE mechanism needs to be trained from scratch based on a new model structure and a large amount of training corpus. The ensemble LLMs require sufficient computing resources to deploy multiple independent LLMs simultaneously.

In this paper, we address these limitations and propose an end-to-end parameter-efficient tuning method designed for multi-task learning on LLMs, dubbed MoA. First, we design a routing mechanism within decoder-only model architecture to automatically select LoRA experts, which can be applied to current mainstream LLMs, and can simultaneously deploy LoRA modules of multiple tasks with limited computing resources (using the same LLM). Meanwhile, our comprehensive model can still achieve excellent performance on different types of tasks. Additionally, to improve the efficiency of training and inference, we implement a parallel processing strategy of different domain samples within a batch during the training process, and a LoRA expert selection approach in the inference time. Our approach leverages the power of different expert models and the base LLM, and the complementary of knowledge in different domains. In summary, our contributions are as follows:

- We propose a MoA architecture for efficient multitask fine-tuning, which can avoid the interference and data imbalance between heterogeneous tasks and easily perform iterative optimization of single task.

- We implement an explicit routing strategy in the training process, which can leverage the knowledge complementarity to further improve the single task performance and ensures the inference efficiency.

- Extensive experiments on various benchmarks are conducted to verify the effectiveness of our approach. Meanwhile, it is flexible to combine multiple domain-specific LoRAs to form a comprehensive LLM.

2 Related Work

Domain Specialization of LLMs. The approaches in LLM domain specialization can be categorized into three corresponding classes of approaches: *external augmentation, prompt crafting, and model fine-tuning* (Zhao et al., 2023). We focus on the third method, which involves updating the LLM's parameters to incorporate domain-specific knowledge directly into the model. Because the current LLMs have billions of parameters and the phenomenon of catastrophic forgetting, we use adapter-based fine-tuning (e.g, Adapter, LoRA) to train multiple domain experts in advance on different task data. LoRA is a parameter-efficient fine-tuning method, which facilitates the adaptation of LLMs using a small-scale external module. As such, LoRA tuning presents a resource-efficient technique to quickly adapt LLMs for novel tasks with restricted training data.

Mixture-of-Experts. The Mixture of Experts (MoE) is an ensemble method, often visualized as a collection of sub-modules, or 'experts', each specializing in processing different types of input data. Each expert is controlled by a router that is then selectively activated based on the type of input data. This technique achieves excellent performance in other domains, including computer vision, speech recognition and multi-modal applications (Fedus et al., 2022a). Inspired by the idea of MoE, we regard each task's LoRA module as a domain expert. Meanwhile, we introduce a routing algorithm to make different domain data automatically choose respective expert, and hence experts in different domains can be combined into a comprehensive model. Furthermore, Shen et al. (2023) also demonstrates that the combination of MoE

and instruction tuning can improve task-specific performance.

Multi-Task Composition. The methods of two-stage learning or end-to-end multi-task learning are commonly used to obtain the combination of multi-task capabilities. The two-stage method requires maintaining a specialized routing model to serve multiple task models, whose overall performance is limited to a single LoRA model. The end-to-end methods introduce new parameter layers or perform implicit parameter fusion. Specifically, Pfeiffer et al. (2020) trains a fusion parameter layer to compose the information stored in the multiple task adapters. Huang et al. (2023) uses a set of learnable weights to integrate multiple LoRA modules into a unified module. Although they can obtain some generalization ability through the combination of parameters trained on various tasks, these fusion methods result in performance degradation on the original task. Zhao et al. (2023) also shows that it is difficult to effectively learn all specialized knowledge of different domains in one LLM.

3 Methodology

The proposed Mixture-of-LoRAs (MoA) can be observed in Figure 2. Given the variations in data scales and training complexities, we commence by separately training N LoRA modules $\{\mathcal{M}_1, ..., \mathcal{M}_N\}$ for N distinct task types $\{\mathcal{T}_1, ..., \mathcal{T}_N\}$. It is worth noting that the task types mentioned here and the domain-specific data or tasks referred to in the paper have the same meaning, all representing scenario tasks that require certain expertise to solve. Initially, we obtain the optimal LoRA module parameters on each scenario task data. These modules demonstrate commendable performance within their respective domains. Subsequently, a routing mechanism is employed to integrate the N LoRA modules under a shared LLM. Specifically, multiple LoRA modules are simultaneously incorporated alongside each transformer layer. Preceding each transformer layer, a routing mechanism is inserted to guide the selection of distinct LoRA experts.

3.1 Learning algorithm

In the first stage, we train a LoRA module for each of the N tasks, which mitigates the problem of catastrophic forgetting of base LLM. Different tasks often have different data scales and training

difficulties. Hence, each LoRA module needs to be optimized individually. These LoRA modules can be arbitrarily combined, added, or removed after initial training. This adapter schemes could enable more fine-grained control over which parts of the LLM are domain-specific.

In the second stage, the N LoRA modules can be aligned with the MoE's expert design. We combine the set of N LoRAs using a routing strategy. While keeping the base LLM parameters Θ fixed, we introduce router parameters $\mathcal{R}$ that learn to select the appropriate expert for users' target tasks (as shown in Figure 2). The trainable router and LoRA parameters are combined to jointly optimize the autoregressive language modeling tasks. The training data for this stage is evenly sampled from the original data of each task, obviating the need to acquire new, high-quality supervised data. The final loss of the MoA is the summation of the language modeling loss and the MoE routing loss:

$$\mathcal{L} = \mathcal{L}_{LM}(\mathbf{x}) + \eta \mathcal{L}_{cls} \qquad (1)$$

Here, η is a parameter that controls the weight of the routing loss and the convergence rate, $\mathcal{L}_{cls}$ is the cross-entropy loss of expert classification, and $\mathcal{L}_{LM}(\mathbf{x})$ is defined as follows:

$$\mathcal{L}_{LM}(\mathbf{x}) = \sum_{i=1}^{n} logP(x_i|x_{<i}) \qquad (2)$$

In this context, the language modeling task (LM) is the commonly used objective for pretraining decoder-only LLMs (e.g., GPT-3 (Brown et al., 2020)). Here, $\mathbf{x}$ represents a sequence of tokens $\{x_1, ..., x_n\}$. The LM task involves predicting the target tokens x_i autoregressively, based on the preceding tokens $x_{<i}$ within the sequence.

By dividing the process into two stages: (1) training domain experts in LoRAs, and (2) combining diverse capabilities through a routing strategy, we effectively address concerns such as catastrophic forgetting, task interference, and instability during multi-task training.

3.2 Routing Strategy

Prior approaches (Fedus et al., 2022b; Lepikhin et al., 2020) on routing strategy have typically focused on learning token-level weighting functions, often assigning one or two experts per token. This approach necessitates careful load balancing to ensure utilization of all experts, prompting the ex-
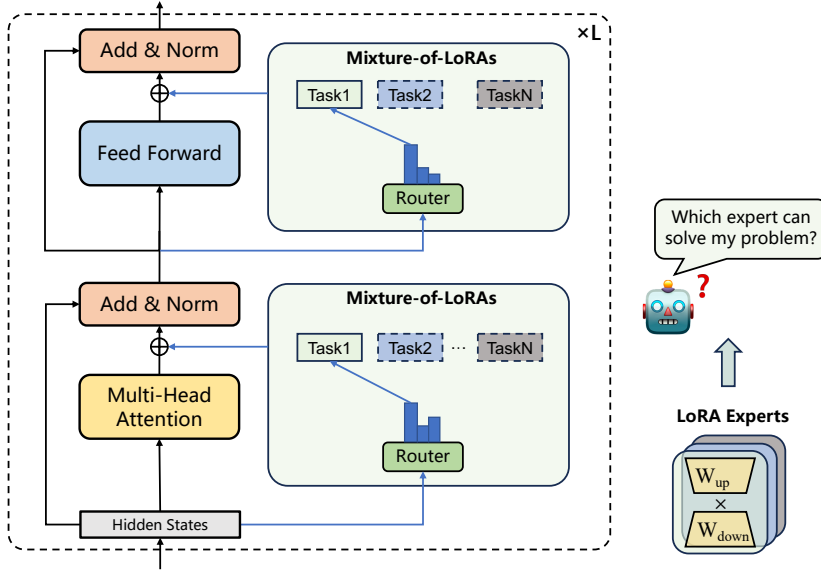
Figure 2: Overall architecture of our proposed MoA.

ploration of explicit balancing mechanisms (Lewis et al., 2021).

In contrast, we adopt a sequence-level routing strategy that leverages domain metadata to route data to LoRA experts. While all token sequences traverse the weight matrices in LLM's transformer layers, during training, each transformer layer employs a distinct router to assign training data labeled with its corresponding domain. The final routing loss enforces precise data-to-expert assignments for each data type. After training with a modest quantity of balanced data, a robust router is obtained. Meanwhile, multi-task training enhances the generalization of the original tasks.

Throughout the training process, language modeling and router classification tasks complement each other. However, we enhance expert selection during inference by employing techniques such as voting or selecting the last expert (as detailed in 4.4). This optimization aims to enhance generation efficiency and contextual consistency.

3.3 MoA Architecture

We design an LoRA expert explicitly for each domain (i.e., eight experts for eight training domains in our multi-domain corpus). LoRA updates weights using the formula:

$$W_0 + \mathbf{router}(W_r, W_1, ..., W_N)$$
$$= W_0 + \mathbf{router}(W_r, A_1B_1, ..., A_NB_N) \quad (3)$$

where $W_0 \in \mathbb{R}^{d \times k}$, $W_r \in \mathbb{R}^{hidden\_dim \times N}$, $A \in \mathbb{R}^{d \times r}$ and $B \in \mathbb{R}^{r \times k}$. W_0 denotes the attention and feed-forward weight matrices of the base

LLM, whose parameters are fixed. The parameter of router($\cdot$) is trainable. W_r is the parameter of the router, which is implemented by a linear layer. The A_iB_i defines the LoRA module $\mathcal{M}_i$, which is repeated multiple times within each transformer layer to reduce trainable parameters for adapting to different domain tasks. The multiple $\mathcal{M}_i$ experts are placed in parallel alongside W_0, departing from previous methods (Pfeiffer et al., 2020; Gururangan et al., 2022; Huang et al., 2023) that add shared fusion layers or replace dense feedforward layers with multiple experts' feedforward networks. The routing algorithm is a key feature in all sparse expert architectures. We adopt an intuitive method to assign different experts to handle tasks in different domains, which helps avoid interference between tasks. Each transformer layer adds a router to select the most appropriate expert. Each router is implemented as a two-layer MLP. This implementation is simple and doesn't significantly increase the number of training parameters.

4 Experiments

4.1 Experimental Setup

Datasets. To evaluate the effectiveness of MoA, we first conduct experiments on various supervised fine-tuning (SFT) datasets of heterogeneous domains. FINANCE, MEDICINE and LEETCODE belong to the specialized domain dataset. EXAM, WEBGPT and GPT4TOOLS limit the output format of the LLM and allow the model to learn special functions. Other datasets include chain-of-

Domain	Source	Language	# Train (Eval.) Tokens
FINANCE	Financial related instructions (Qingyi Si, 2023)	EN	1.2M (0.24M)
MEDICINE	10k real conversations between patients and doctors (Li et al., 2023)	EN	1.4M (0.28M)
LEETCODE	Chinese Open Instruction Generalist (Zhang et al., 2023)	CN	9.3M (2.09M)
EXAM		CN	3.6M (0.71M)
WEBGPT	Retrieval question answering dataset (Nakano et al., 2021)	EN	7.4M (1.46M)
GPT4TOOLS	A collection of tool-related instructions (Yang et al., 2023)	EN	7.5M (1.49M)
COT	Several Chain-of-Thought datasets (Longpre et al., 2023)	EN	1.1M (0.22M)
STACKOVERFLOW	57k dialogs from StackOverFlow questions (Xu et al., 2023)	EN	0.9M (0.18M)

Table 1: Statistics of SFT datasets. # Train (Eval.) Tokens denotes the size of our training and evaluation (i.e. validation and test) data, obtained via the Qwen tokenizer. We evenly sample about 10k training data and 2k validation data on each dataset.

thought, dialog, etc. Meanwhile, both English and Chinese are involved. We show the statistics of all datasets in Table 1.

Implementation Details. We use the *Qwen-7b* as our base LLM, which is the decoder-only architecture. We use the vocabulary of 151,851 BPE types, and train with 4,096-token sequences. We set the total number of training steps based on this allocated runtime, set 10% of these steps to be warmup, and use the AdamW (Loshchilov and Hutter, 2017) optimizer with a cosine learning rate decay. Learning rate is set to $1e^{-5}$. Each worker processes eight sequences of length 4,096, and gradients are accumulated over 4 updates. We clip gradients if their L2 norm exceeds 0.1. In the inference time, we report test perplexity after a single run of training on 8 NVIDIA A100 80GB GPUs.

4.2 Models and Metrics

Single-LoRA The first baseline is a LoRA trained on data within the domain, which could be viewed as a specialized model for handling domain tasks.

Single-LoRA (mixed) We train a single LoRA on mixed data from all domains. While there is no explicit conditioning on domain, the gradient updates during training average across all domains represented in a batch.

MoA We add multiple domain LoRAs in the transformer based on the first baselines, and use routing strategy and domain label information for multitask training. Under the *routing* setting, the test data domain is unknown.

MoE-LoRA We leverage the idea of sparsely activated Mixture-of-Experts (MoE). We add multiple domain LoRA modules on the bypass of transformer layer. Each input token will be processed by a limited subset of experts. Different from our MoA, this approach can add any number of ex-

perts and has no concept of data domain.

MoE-LoRA (naive) The architecture of this model is exactly the same as MoE-LoRA, and the only difference is that we randomly initialize all LoRA modules.

To comprehensively evaluate the performance of different models, we use several evaluation metrics, including the perplexity (PPL) of generated texts, and the bilingual evaluation understudy (BLUE) and the longest common subsequence (ROUGE-L) between the generated answer and the gold answer.

4.3 Main results

Classifier+LoRAs The most intuitive method of integrating multiple LoRA experts is to use a specific classifier to act as a distributor, as shown in Figure 3. We train a classifier based on roberta-base (Liu et al., 2020) using the same training data as MoA. The specific classification performance is shown in Table 3. This approach is so flexible that we can combine multiple LoRA modules and avoid interference between tasks. However, the performance of this approach is limited to each LoRA expert. From the classification results, our router performs better than the classifier overall.
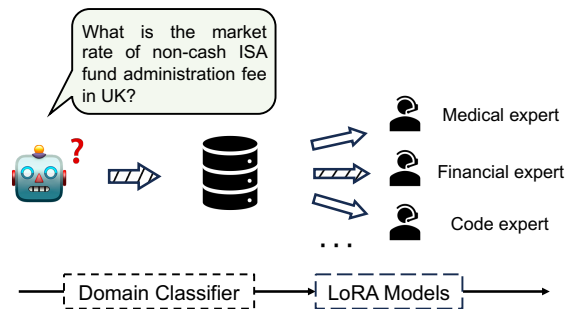

Figure 3: The flow diagram of classifier+LoRAs.

End-to-end methods The end-to-end methods mean that one model can directly solve multi-

Domain	Single-LoRA			Single-LoRA (mixed)			MoA		
	PPL	BLUE	ROUGE-L	PPL	BLUE	ROUGE-L	PPL	BLUE	ROUGE-L
FINANCE	7.8479	18.5975	28.6266	7.7214	**22.4846**	**32.5574**	**7.5287**	20.5774	30.6797
MEDICINE	9.5097	13.6096	18.8911	9.0499	13.5373	19.4425	**8.4561**	**13.8811**	**19.8118**
LEETCODE	1.9527	34.8582	47.8152	2.0289	35.2886	46.6290	**1.9311**	**37.4872**	**49.3256**
EXAM	3.1154	3.0871	18.5609	3.1135	4.3259	16.6206	**2.9752**	**4.7942**	**19.1840**
WEBGPT	1.7945	38.8995	41.4447	1.8484	39.6297	42.0700	**1.7933**	**40.2602**	**43.7395**
GPT4TOOLS	2.2525	64.7501	71.4391	2.2497	66.3450	73.1289	**2.2123**	**69.2596**	**74.5962**
COT	2.8126	34.5210	45.7961	2.6474	43.6290	**53.2125**	**2.5931**	40.2529	50.3844
S.O.	**2.8169**	19.9554	29.7282	2.9012	19.4896	28.4694	2.8999	**23.0412**	**31.9793**
Average	4.0128	28.5348	37.7877	3.9450	30.5912	39.0163	**3.7987**	**31.1942**	**39.9626**

Table 2: In-domain test-set performance for different training strategies of LoRA. We report PPL, tokenized BLUE and ROUGE-L above. Both Single-LoRA (mixed) and MoA are tested under the test data of unknown domain labels and Single-LoRA is tested on the test data of the corresponding domain. PPL is short for perplexity. The best value per metric on different tasks is in bold.

Domain	test size	Classifier	Router
FINANCE	2000	98.80%	99.60%
MEDICINE	1221	99.92%	99.92%
LEETCODE	1952	99.95%	100.00%
EXAM	1999	99.95%	100.00%
WEBGPT	2000	100.00%	99.85%
GPT4TOOLS	2000	100.00%	100.00%
COT	2000	99.75%	99.95%
STACKOVERFLOW	2000	99.00%	99.90%
Average	1896.5	99.67%	**99.90%**

Table 3: The classification accuracy of MoA router and a specific classifier by domain at inference time.

Model	LoRA	LoRA (mixed)	MoA
trainable parameters	143M	143M	143M*8+1.05M

Table 4: The trainable parameters under different LoRA combinations. The router module takes up only 1.05M parameters.

ple tasks even if the test data domain is unknown. Table 2 shows that test perplexities, blue-4 and rouge-l, averaged across the eight training domains. Training in the mixed domain data is helpful for the overall performance (Perplexity: 4.0128→3.9450, BLUE: 28.5348→30.5912, ROUGE-L: 37.7877→39.0163). However, the performance decreases on data with strict output formats such as WEBGPT, STACKOVERFLOW, which also shows that not all additional domain information is complementary. We hypothesize that separate training is advantageous for heterogeneous domains. Therefore, we design an efficient multi-task learning method to avoid interference between partial tasks. Our approach achieves significant improvements across all datasets.

In order to further validate the reliability of the MoA's performance, we conducts accuracy evaluation experiments on datasets with standard answers. The EXAM dataset is utilized for the experiments, as it primarily consists of secondary and university entrance exam questions, with the majority being multiple-choice questions (including

single and multiple selections) and a small portion of true/false questions. Accuracy is calculated by directly comparing the answers with the reference answers. Specifically, string processing functions, regular expressions, etc., are used to parse out specific options (A/B/C/D) or judgment results (T/F) from the answers, followed by calculating the accuracy of responses. The final results are presented in Table 5. Despite the overall low accuracy due to the difficulty of the questions, the accuracy of MoA is significantly higher than the other two models (**+5.86%**, **+5.48%**).

Model	Total	Right	Accuracy
Single-LoRA (mixed)	1331	515	38.69%
Single-LoRA	1331	520	39.07%
MoA	1331	593	**44.55%**

Table 5: The accuracy of responses on the Exam test dataset.

In addition, this subsection also introduces GPT-4 as an evaluation expert to assess FINANCE, MEDICINE, and WEBGPT datasets. As the answers in these datasets do not follow a fixed pattern, this subsection adopts the common evaluation method in the community of LLMs. Evaluation scoring is conducted through a larger model.

In this study, GPT-4 is utilized to provide accuracy scores for the model's responses to questions and standard answers. The complete evaluation prompt is illustrated in Appendix A.

Apart from the three models: Single-LoRA, Single-LoRA (mixed), and MoA, we also evaluates the individual LoRA modules within the MoA model. These are represented in the table 6 as Single-LoRA of MoA. Due to fluctuations in scoring by the large language model, each scoring process is invoked three times and the average is taken. From the experimental results, it is observed that after multi-task learning training within MoA, the performance of each LoRA module surpasses the original Single-LoRA modules in each task. When handling specific professional tasks individually, these modules exhibit outstanding performance. Therefore, through multi-task learning methods, the performance of multiple LoRA modules is further enhanced while effectively collaborating.

Score Dataset Model	Finance	Medicine	Webgpt
Single-LoRA (mixed)	**76.91**	57.49	87.92
Single-LoRA	75.99	57.11	88.59
Single-LoRA of MoA	76.30	<u>58.01</u>	<u>89.00</u>
MoA	<u>76.56</u>	**60.68**	**89.27**

Table 6: The evaluation scoring of the GPT-4 on the Finance, Medicine, and WebGPT datasets. The highest value per column is in bold and the second highest value is <u>underlined</u>.

Based on the metrics such as PPL, BLUE, ROUGE, and accuracy metrics for specific downstream tasks presented in the paper, we can conclude that our proposed simple and efficient architecture effectively enables learning of various domain-specific competencies within a single large language model, while also avoiding interference between different tasks. Furthermore, each functional module is relatively independent, facilitating efficient consolidation of additional data for further optimization. Additionally, this method significantly saves computational resources during deployment.

4.4 Mixing LoRA Experts at Inference Time

The previous section establishes that our multi-task training method improves the performance of the single LoRA expert on test data. In addition, the mixing of multi-domain data is effective in the

training process. In practice, however, the original training data scale of the respective tasks is relatively large and textual data to be evaluated may not come with a domain label.

In these cases, the mixed training of a large amount of data is unfavorable for our later iterations of single tasks. We propose to treat $LoRA_1, ..., LoRA_N$ as domain experts, transforming the input text into a matter of expert selection. The routing strategy is introducing to solve the problem of interference between tasks. The parameter of router is shown in Table 4, which only accounts for a very small percentage. Our approach is parameter-efficient and selective complementarity between tasks. The results of evaluation on the test data of unknown domain are shown in Table 2.

To further explore the actual effect of the routing module, we test its selection ability in the inference process. The number of our routers is consistent with the number of transformer layers in our LLM. Each router is implemented with a linear layer. Therefore, to improve the efficiency of the inference process and the consistency of generation, we choose to use (1) voting of all routers and (2) the result of last router as the selected LoRA module in the subsequent inference process. The experimental results show that the classification performance of the last router is more stable. Finally, we only use 1.05M routing parameters to achieve an average accuracy of 99.90% in Table 3. Overall, the router plays two key roles in our model architecture. One is to learn the complementarity of non-heterogeneous domain knowledge and in the training process. The other is to select appropriate LoRA expert to solve problems in unknown domains in the inference process.

Methods	PPL	BLUE	ROUGE-L
MoE-LoRA	3.8578	29.1640	37.5960
MoE-LoRA (naive)	**3.7969**	29.4170	37.3917
MoA	3.7987	**31.1942**	**39.9626**

Table 7: The averaged test performance comparison on eight tasks. All of the above methods have the same number of LoRA experts and have exactly the same model parameters.

4.5 Ablation Studies

We further conduct specific experiments to investigate the effectiveness of different components in

Query: There are 14 chickens in the coop, and twice that many in the run. There is 4 less than double the number of chickens free ranging as in the run. How many chickens are free ranging? Let's be accurate as possible. Gold Answer: Coop:14 chickens. Run:14(2) = 28 chickens. Free Range:28(2) - 4 = 52 chickens. The answer: 52.	Single-LoRA ✗	There are 14 * 2 = 28 chickens in the run. There are 28 - 4 = 24 chickens free ranging. The answer: 24.
	Single-LoRA (mixed) ✗	
	MoE-LoRA ✗	There are 14 * 2 = 28 chickens free ranging. The total number of chickens is 14 + 28 = 42. The answer: 42.
	MoE-LoRA (naive) ✗	The number of chickens free ranging is 2 * 14 = 28. There are 28 - 4 = 24 chickens in the coop. Thus, there are 14 + 24 = 38 chickens in total.
	MoA √	The total number of chickens in the coop and run is 14 + 14 = 28. Double the number of chickens in the run is 28 * 2 = 56. There are 56 - 4 = 52 chickens free ranging. Final answer: 52.

Figure 4: Case study of the predicted output of different models.

our model.

Impact of Domain Label Information In addition to the language modeling loss, we also add the domain label routing loss in the training process. To verify the effectiveness of our proposed mechanism, we remove the routing loss and our model devolved into MoE-LoRA. Therefore, the MoE-LoRA does not introduce explicit domain label information in the training and inference process and guarantees the same number of parameters as MoA. As shown in Table 7, MoA has achieved an overall improvement over MoE-LoRA, which demonstrates that the domain label information is useful for different tasks. Furthermore, we evaluate the performance of single LoRA module after multi-LoRA joint training in Table 8. The MoA further improves the perplexity performance of each LoRA module from 4.0128 to 3.7962, while MoE-LoRA causes a slight decrease. Therefore, when we need to expand to more tasks or combine multiple functions, MoA is more flexible and effective.

Domain	Single-LoRA	MoE-LoRA	MoA
FINANCE	7.8479	7.6623	**7.5235**
MEDICINE	9.5097	9.6510	**8.4488**
LEETCODE	1.9527	2.0087	**1.9296**
EXAM	3.1154	3.1455	**2.9745**
WEBGPT	1.7945	1.8080	**1.7927**
GPT4TOOLS	2.2525	2.2524	**2.2123**
COT	2.8126	2.9205	**2.5910**
S.O.	**2.8169**	2.8801	2.8968
Average	4.0128	4.0411	**3.7962**

Table 8: The test perplexity of corresponding LoRA module in different models on each task dataset.

Impact of Different Initialization Methods To evaluate the effectiveness of various LoRA initialization methods, we present the MoE-LoRA (naive) approach. In contrast to MoE-LoRA, the LoRA modules in the naive variant are initialized randomly. This approach leverages the complete dataset from all domains, leading to a significant increase in total training duration. However, this approach proves to be highly inefficient when we require customization of different capability combinations. Moreover, the comparative analysis presented in Table 7 demonstrates that both methods yield comparable results across eight tasks. Therefore, conducting further training on the initial multi-domain LoRA parameters emerges as a highly efficient method.

4.6 Case Study

Figure 4 shows the comparison of the outputs of different models on a specific reasoning question. From the above results, only the MoA correctly understands the multiple relationships in the query and predicts the right answer, while other approaches are slightly less capable of reasoning. Furthermore, the MoA is superior to the single-LoRA, which also illustrates the advantages of our multitask fine-tuning. This approach can not only avoid the interference between different tasks, but also further improve the performance on a single task, so it has a great application prospect.

5 Conclusions

We introduce MoA architecture, which provide an efficient multi-task fine-tuning method for LLM, addressing interference among tasks and training instabilities. Each LoRA model can be iterated individually to quickly adapt to new domains. Meanwhile, MoA uses routing strategy to flexibly select the appropriate LoRA expert to solve

the problem. It can arbitrarily combine multiple domain-specific LoRA modules to implement a LLM with multiple specific capabilities. Future work may focus on how to flexibility add or remove LoRA modules with unsupervised learning, optimize the current routing algorithm, or reduce the scale of training data in domain specialization of LLMs.

References

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, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey 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. 2020. Language models are few-shot learners.

Jiaxi Cui, Zongjian Li, Yang Yan, Bohua Chen, and Li Yuan. 2023. Chatlaw: Open-source legal large language model with integrated external knowledge bases.

William Fedus, Jeff Dean, and Barret Zoph. 2022a. A review of sparse expert models in deep learning. *arXiv preprint arXiv:2209.01667*.

William Fedus, Barret Zoph, and Noam Shazeer. 2022b. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270.

Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah A. Smith, and Luke Zettlemoyer. 2022. DEMix layers: Disentangling domains for modular language modeling. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5557–5576, Seattle, United States. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

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

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

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. Llm-blender: Ensembling large language models with pairwise comparison and generative fusion. In *Proceedings of the 61th Annual Meeting of the Association for Computational Linguistics (ACL 2023)*.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. 2021. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, pages 6265–6274. PMLR.

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

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Ro{bert}a: A robustly optimized {bert} pretraining approach.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2021. Webgpt: Browser-assisted question-answering with human feedback. In *arXiv*.

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

Zheng Lin Qingyi Si. 2023. Alpaca-cot: An instruction fine-tuning platform with instruction data collection and unified large language models interface. https://github.com/PhoebusSi/alpaca-CoT.

Sheng Shen, Le Hou, Yanqi Zhou, Nan Du, Shayne Longpre, Jason Wei, Hyung Won Chung, Barret Zoph, William Fedus, Xinyun Chen, Tu Vu, Yuexin Wu, Wuyang Chen, Albert Webson, Yunxuan Li, Vincent Zhao, Hongkun Yu, Kurt Keutzer, Trevor Darrell, and Denny Zhou. 2023. Mixture-of-experts meets instruction tuning:a winning combination for large language models.

Ruoxi Sun, Sercan O. Arik, Hootan Nakhost, Hanjun Dai, Rajarishi Sinha, Pengcheng Yin, and Tomas Pfister. 2023. Sql-palm: Improved large language model adaptation for text-to-sql.

Bhaskar Tripathi. 2023. Pdf-gpt. https://github.com/bhaskatripathi/pdfGPT.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. Bloomberggpt: A large language model for finance.

Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. 2023. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. *arXiv preprint arXiv:2304.01196*.

Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. 2023. Gpt4tools: Teaching llm to use tools via self-instruction.

Ge Zhang, Yemin Shi, Ruibo Liu, Ruibin Yuan, Yizhi Li, Siwei Dong, Yu Shu, Zhaoqun Li, Zekun Wang, Chenghua Lin, Wenhao Huang, and Jie Fu. 2023. Chinese open instruction generalist: A preliminary release.

Xujiang Zhao, Jiaying Lu, Chengyuan Deng, Can Zheng, Junxiang Wang, Tanmoy Chowdhury, Li Yun, Hejie Cui, Zhang Xuchao, Tianjiao Zhao, et al. 2023. Domain specialization as the key to make large language models disruptive: A comprehensive survey. *arXiv preprint arXiv:2305.18703*.

A Evaluation prompt

The complete prompt used for invoking the GPT-4 API is as follows:

eval_prompt = """ You will receive three parts of content: the questioner's question, the user's answer, and the reference answer.

Your task is to score the accuracy of the user's answer based on the following criteria. Please ensure that you read and understand these instructions carefully.

Evaluation Criteria:

Accuracy - Whether the user's answer is consistent with the reference answer and has addressed the questioner's question. We define this dimension as 'whether the user's answer includes all the key points from the reference answer and has addressed the questioner's question.'

Evaluation Steps:

1. Carefully read the questioner's question, understand the key points of the question.

2. Carefully read the reference answer, understand the key points related to the question contained in the reference answer.

3. Check if the user's answer includes the key points from the reference answer and has addressed the questioner's question.

4. Based on the evaluation criteria, score within a range of 0 to 100, where 0 means the user's answer does not contain any key points from the reference answer and has completely failed to address the questioner's question; 100 means the user's answer includes all the key points from the reference answer and has correctly and completely addressed the questioner's question.

Example: Questioner's question: [query] User's answer: [llm_answer] Reference answer: [fact]

Evaluation result (score only):

Accuracy (0-100): """

where "[query]" represents user input, "[llm_answer]" denotes the output of the MoA model, and "[fact]" stands for the standard answer provided by the dataset.