Merging LoRAs for language model's multitask ability

UNIST CSE402 Custom Project

Chanbin Lee
Department of Computer Science & Engineering
UNIST
chblee@unist.ac.kr

1 Research paper summary (max 2 pages)

- **Title** Mixture-of-LoRAs: An Efficient Multitask Tuning for Large Language Models
- **Venue** Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING)
- **Year** 2024
- **URL** https://aclanthology.org/2024.lrec-main.994/

Background. Large Language Models (LLMs) have transformed NLP applications through their strong generalization capabilities. However, real-world usage often demands domain-specific expertise, such as financial analysis, legal reasoning, or medical consultations. Fine-tuning large models for multiple specialized tasks presents critical challenges: (1) Catastrophic forgetting where fine-tuning for one task causes performance degradation on others; (2) Resource inefficiency since full-model fine-tuning is computationally expensive; and (3) Task interference when naive multi-task training mixes data from very different domains, harming specialization.

Previous efforts to address these problems include AdapterFusion, Mixture-of-Experts (MoE) models. However, implicit parameter fusion (e.g., AdapterFusion) often causes interference between domains, while MoE models typically require expensive retraining from scratch and large compute resources. These methods struggle to balance flexibility, performance, and resource efficiency.

Low-Rank Adaptation (LoRA) is a parameter-efficient fine-tuning technique proposed to adapt large pre-trained models to downstream tasks without updating all model parameters. Instead of modifying the original weight matrices directly, LoRA introduces trainable low-rank updates while freezing the original weights.

Given a weight matrix $W_0 \in \mathbb{R}^{d\times k}$ in a pre-trained model, LoRA decomposes the update into two smaller matrices:

$$\Delta W = AB$$

where $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times k}$, and $r \ll \min(d, k)$ is the rank of the decomposition.

The updated weight during fine-tuning becomes:

$$W = W_0 + \Delta W = W_0 + AB$$

Here, only A and B are learned during fine-tuning, while W_0 remains frozen. LoRA significantly reduces the number of trainable parameters by enforcing a low-rank structure, making fine-tuning more memory-efficient and computationally affordable, especially for large language models (LLMs).

When LoRAs are trained for individual downstream tasks, they often perform as well as full-finetuning. If the model can dynamically use these separately trained LoRAs for the task given, then the model will have ability to handle all the downstream tasks well.

Summary of contributions. The paper introduces **Mixture-of-LoRAs (MoA)**, a novel architecture for efficient multitask fine-tuning of large language models (LLMs). Instead of conventional approaches that mix training data or jointly fine-tune shared parameters, MoA modularizes expertise by training domain-specific **Low-Rank Adaptation (LoRA)** modules independently, then combining them through an explicit **routing strategy**. The method consists of two major stages:

First, for each domain or task T_i, a separate LoRA module M_i is trained using supervised datasets corresponding to that domain. LoRA modules are lightweight adapters inserted into the LLM's attention and feed-forward layers, consisting of low-rank matrices $A_i \in \mathbb{R}^{d\times r}$ and $B_i \in \mathbb{R}^{r\times k}$, where $r \ll d, k$. The base model weights W_0 are frozen during this process, and only the LoRA parameters are updated. Each LoRA learns highly domain-specific knowledge without causing catastrophic forgetting.

After training LoRA experts individually, they are combined under a shared base model. A **router network** is introduced before each transformer block to select the appropriate LoRA module for each input sequence. The router is optimized jointly with the LoRA modules during training using a combined loss:

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

where $\mathcal{L}_{LM}$ is the language modeling loss, and $\mathcal{L}_{cls}$ is a cross-entropy loss enforcing correct domain classification.

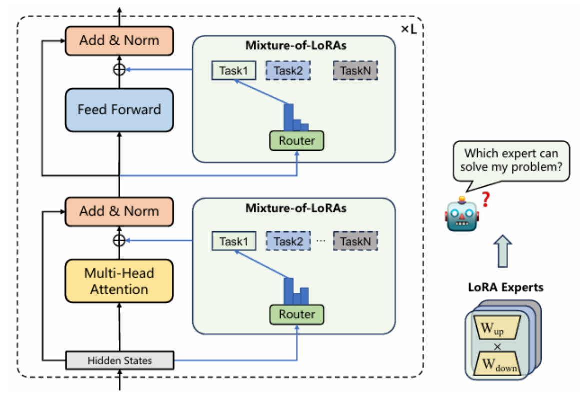

Figure 2: Overall architecture of our proposed MoA.

Figure 1: How MoA works.

Routing Strategy

Unlike prior Mixture-of-Experts (MoE) methods which route at the token level, MoA's routing operates at the **sequence level**, using domain metadata to guide selection. During inference, routers can either *vote* across layers or use the decision from the last router to select the LoRA expert. This approach avoids task interference and improves generation consistency.

Efficient Multitask Inference

At inference time, only the selected domain-specific LoRA is activated per input, minimizing computational overhead. MoA enables a single LLM to support multiple specialized capabilities simultaneously, covering heterogeneous domains like finance, medical conversations, code generation, and more.

Superior Experimental Results

Experiments across eight diverse benchmarks (FINANCE, MEDICINE, LEETCODE, EXAM, WEBGPT, GPT4TOOLS, COT, STACKOVERFLOW) demonstrate that MoA outperforms baselines such as Single-LoRA, mixed-domain LoRA, and naive MoE-LoRA variants. MoA achieves better perplexity (PPL), BLEU, and ROUGE-L scores across all tasks.

Flexibility and Scalability

Because LoRA modules are independently trained, new domain expertise can be added by simply training a new LoRA and integrating it through the routing mechanism without retraining existing modules. This modular design makes MoA highly scalable and suitable for continual learning and real-world deployment scenarios.

Limitations and discussion. While MoA shows strong performance, it assumes the availability of clear domain metadata for routing during training. In fully unsupervised settings, this reliance on domain labels could limit its direct applicability. Moreover, although routing overhead is minimal (around 1M parameters), there remains a slight complexity increase compared to fully static LoRA compositions. Finally, the assumption that domains are separable may not hold in highly entangled datasets where task boundaries are ambiguous.

The paper also leaves open the question of optimizing routing mechanisms further (e.g., using unsupervised clustering or continual learning settings), suggesting this as a promising direction for future work.

Why this paper? This paper was chosen because it presents a practical solution to a fundamental problem in LLM adaptation: efficiently specializing large models for multiple domains without catastrophic forgetting or excessive retraining cost. The modularity and efficiency of MoA align well with broader trends in scalable, real-world deployment of LLMs. Moreover, the combination of explicit routing and LoRA-based adaptation is simple but innovative, and it seemed to be able to be extended in various ways.

Wider research context. MoA fits naturally within the broader landscape of research on parameter-efficient fine-tuning (PEFT) and multitask learning for LLMs. In the context of NLP, it addresses core challenges such as catastrophic forgetting, task interference, and domain generalization. Compared to prior work such as AdapterFusion (Pfeiffer et al., 2020), MoA provides a cleaner, more flexible alternative that better handles domain heterogeneity.

2 Project description (1-2 pages)

Goal. The paper showed that their method, MoA (Mixture of Adapters), is effective and performs well across multiple tasks. Their approach involves training separate LoRA modules (Low-Rank Adaptation modules) for each task and subsequently training a router to dynamically select the appropriate LoRA weights during inference. However, this setup introduces additional complexity: after fine-tuning LoRAs individually, an extra step of router training is necessary.

This raises an question: **If the LoRAs were trained on completely different tasks, with clearly distinct input structures, is a MoA-like router system still necessary?** Could simpler techniques, such as directly merging LoRA weights—through concatenation, weighted sums, or other methods—be sufficient? In other words, could a well-trained language model inherently leverage contextual information to activate the appropriate parts of merged LoRA modules without explicit routing?

I aim to explore this hypothesis by first implementing a system similar to the Mixture-of-LoRAs (MoA) architecture as proposed in the paper. After completing the implementation, I will conduct a comparative evaluation between the MoA method and several custom-designed, simpler baseline methods. While it is expected that the MoA-style system will deliver strong performance (given its specialized routing mechanism), I will investigate whether the simpler methods can achieve comparable results, thus validating the potential of a more streamlined, resource-efficient approach.

Task. To comprehensively test this hypothesis, I plan to evaluate across a broad range of natural language processing (NLP) tasks that differ substantially in nature and difficulty, including:

- Summarization
- Sentiment Analysis
- Translation
- Question Answering (QA)
- Dialogue/Chatting
- Reasoning/Math Problem Solving

Data. The datasets selected are all standard, widely-used benchmarks that offer enough data and established evaluation protocols:

- **Summarization**: `CNN/DailyMail` – approximately 287,000 news articles and summaries.
- **Translation**: `WMT14 En-De` (English-German translation) – around 4.5 million sentence pairs.
- **Question Answering**: `SQuAD v1.1` – over 100,000 question-answer pairs based on Wikipedia articles.
- **Dialogue/Chatting**: `OpenOrca` or `ShareGPT` – several million user-assistant conversations (processed for instruction tuning).
- **Reasoning**: `GSM8K` – approximately 8,500 grade-school math word problems with detailed solutions.

All datasets will be preprocessed to ensure inputs are distinguishable enough to help the model recognize task types based solely on the context.

Methods. I will build on open-source 7B-scale language models like **LLaMA** or **Mistral**, ensuring compatibility with flexible LoRA insertion. Since MoA requires architectural changes—specifically the addition of a router—I will avoid using Huggingface's prepackaged model classes, opting instead to work directly with the original PyTorch model implementations.

The following components will be implemented:

- Custom LoRA modules
- Router mechanism (for MoA)
- Baseline mechanisms for LoRA merging without dynamic routing

All implementations, including the baselines, will be developed from scratch to ensure consistent experimental control.

Baselines. Several simple baseline methods will be constructed for comparison:

- **Concatenated LoRAs**: Simply stacking LoRA modules and feeding the outputs without selective routing.
- **Weighted Sum Merge**: Pre-defining static weight combinations for the LoRA modules and merging them linearly.
- **SVD Merge**: Applying singular value decomposition (SVD) to compress multiple LoRA weights into a single low-rank matrix.

Each baseline is designed to test whether routing complexity is truly necessary, or if simpler merging methods could suffice under well-separated task conditions.

Evaluation. Performance will be assessed using standard task-specific metrics:

- **Summarization** (`CNN/DailyMail`): ROUGE-1, ROUGE-2, ROUGE-L scores.
- **Translation** (`WMT14 En-De`): BLEU score.
- **Question Answering** (`SQuAD v1.1`): Exact Match (EM) and F1 scores.
- **Dialogue/Chatting** (`OpenOrca` or `ShareGPT`): Perplexity (PPL) and optionally human evaluation.
- **Reasoning/Math** (`GSM8K`): Exact Match (EM) accuracy on final numerical answers.

Each model variant—MoA, Concatenated LoRAs, Weighted Merge, and SVD Merge—will be evaluated uniformly across all tasks. Beyond pure metrics, I will also observe qualitative behavior of the models, such as whether they correctly identify the task type from input formatting cues without explicit routing guidance.

3 References

Mixture-of-LoRAs: An Efficient Multitask Tuning Method for Large Language Models (Feng et al., LREC-COLING 2024)