Rethinking LoRA Merging for Language Model's Multitask Adaptation

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

Abstract

Language models are increasingly adapted for multitask applications through parameter-efficient fine-tuning techniques such as Low-Rank Adaptation (LoRA). Mixture-of-LoRAs (MoA) introduces a routing-based mechanism to activate task-specific LoRA modules. However, the necessity of such routing remains questionable when task inputs are clearly distinguishable. In this work, we explore the viability of routing-free alternatives by merging LoRA modules using simple strategies including concatenation, weight summing, and singular value decomposition (SVD). Empirical evaluations across standard NLP tasks demonstrate that merged LoRAs can approach or match MoA performance under structured prompting, suggesting that routing mechanisms may be avoidable in certain scenarios.

1 Introduction

Language models have become the backbone of modern natural language processing (NLP), achieving state-of-the-art results across a wide range of tasks, including summarization, translation, question answering, and more. However, adapting these models to multiple downstream tasks in a resource-efficient and scalable manner remains a key challenge. Traditional full fine-tuning of language models for each individual task is computationally prohibitive and introduces the risk of *catastrophic forgetting*, where performance on previously learned tasks degrades as new tasks are introduced.

To address these limitations, a growing body of work has focused on parameter-efficient fine-tuning (PEFT) techniques. Among these, Low-Rank Adaptation (LoRA) has emerged as a popular solution due to its ability to fine-tune only a small number of parameters while keeping the base model frozen. By injecting low-rank trainable matrices into existing weight layers, LoRA allows for task-specific adaptation without retraining the entire model. This modularity makes LoRA an appealing building block for multitask learning.

Despite its efficiency, naively deploying multiple LoRA modules—each fine-tuned on a separate task—raises a practical question: how can we effectively combine or coordinate these modules at inference time? One recent proposal, Mixture-of-LoRAs (MoA), addresses this by introducing a learned routing mechanism that dynamically selects the appropriate LoRA module based on the input. This design achieves strong multitask performance by leveraging explicit task-level routing, but it also introduces architectural complexity and additional training overhead.

This work builds on the MoA framework and seeks to explore a key open question: **Is routing truly necessary when tasks are well-separated and easily distinguishable via structured input prompts?** Intuitively, if each task's input format is sufficiently unique, it may be possible for a language model to infer the appropriate behavior based solely on context, even when LoRA modules are merged rather than explicitly routed.

To this end, we evaluate the performance of various routing-free alternatives to MoA, including simple merging strategies such as LoRA concatenation, weight summing, and singular value decomposition (SVD). These methods eliminate the need for a router while preserving the benefits of modular adaptation. We implement both the MoA architecture and these baselines, and conduct experiments on diverse NLP tasks.

Our results show that while MoA generally performs best, routing-free methods achieve competitive or even superior performance on certain tasks. These findings suggest that for well-separated tasks with structured prompts, merging-based approaches can serve as viable, low-overhead alternatives to routing.

This work makes the following contributions:

- We propose and compare multiple routing-free LoRA merging strategies under a shared multitask setup.
- We conduct comprehensive experiments on diverse NLP tasks and report the task performances.
- We demonstrate that well-structured prompts can enable effective multitask inference without routing.

2 Related Work

2.1 Parameter-Efficient Fine-Tuning

Traditional fine-tuning of language models involves updating all model parameters for each new task, which is computationally expensive and prone to catastrophic forgetting (Goodfellow et al., 2014). To address these issues, *parameter-efficient fine-tuning* (PEFT) techniques have been proposed. One of the earliest and most influential approaches in this area is Adapter Tuning (Houlsby et al., 2019), which inserts small bottleneck layers between existing transformer layers, fine-tuning only the adapters while keeping the base model frozen.

LoRA (Low-Rank Adaptation) (Hu et al., 2022) improves on this by injecting trainable low-rank matrices into existing weight layers (e.g., query and value projections in attention modules). It achieves comparable performance to full fine-tuning while introducing only a small number of task-specific parameters. Other notable PEFT methods include BitFit (Ben Zaken et al., 2022), which fine-tunes only bias terms, and Prefix Tuning (Li and Liang, 2021), which optimizes continuous task-specific prompts.

These methods enable scalable fine-tuning for multiple tasks or domains by decoupling adaptation from the full model. However, they do not directly address how multiple task-specific modules should be integrated during inference in a multitask setting.

2.2 Multitask Adaptation and Mixture-of-Experts

Multitask learning for language models faces the challenge of task interference, where training on multiple tasks jointly can lead to degraded performance due to conflicting optimization signals (Ruder, 2017). One class of solutions is Mixture-of-Experts (MoE) models (Shazeer et al., 2017), which introduce multiple expert modules and use a gating mechanism to dynamically select a subset for each input. While effective, MoE models typically operate at the token level and require expensive joint training and expert balancing strategies (Lepikhin et al., 2021).

The Mixture-of-LoRAs (MoA) framework (Feng et al., 2024) builds on these ideas by using LoRA modules as lightweight experts. Instead of token-level gating, MoA performs routing at the sequence level, selecting a single LoRA module per input sequence. This greatly reduces routing overhead and allows independently trained LoRA modules to be reused. During training, a router is learned by constructing a dataset where each input is paired with its task label, enabling supervised classification of task type. This approach achieves strong results across diverse domains while maintaining modularity and scalability.

However, MoA assumes that task metadata or clear domain labels are available, and requires additional router training after LoRA modules are trained. This motivates our exploration of routing-free alternatives in scenarios where tasks are well-separated and inputs are distinguishable.

2.3 Model Merging and Composition

Recent research has also explored model merging techniques as a way to combine multiple task-specific models or adapters without requiring explicit routing. Interpolation-based methods (Wortsman et al., 2022) linearly combine weights from different models, while *SVD merging* (Ilharco et al., 2023) compresses multiple models into a single low-rank representation.

In the context of adapters, AdapterFusion (Pfeiffer et al., 2020) combines multiple pre-trained adapters through attention-based fusion. However, this method requires joint fine-tuning of the fusion layer and often suffers from interference when tasks are not clearly separated.

To the best of our knowledge, little work has explored direct merging of LoRA modules in multitask settings without additional routing or fusion networks. This work contributes to this line of research by systematically evaluating the performance of simple merging strategies (e.g., concatenation, weight summing, SVD) as alternatives to routing-based approaches like MoA.

2.4 Prompt-Based Task Conditioning

Recent advances in instruction tuning (Wang et al., 2022; Sanh et al., 2022) have demonstrated that language models can learn to generalize across tasks when provided with clearly structured prompts. Models such as T0 (Sanh et al., 2022) and FLAN (Wei et al., 2022) show that prompt-based input formatting can serve as a powerful task identifier, even in zero-shot settings.

In this work, we leverage this insight by using structured prompts to condition each task. Our hypothesis is that when tasks are sufficiently distinguishable through prompt formatting, a merged LoRA module may implicitly activate task-relevant behavior without the need for an explicit router.

3 Approach

We present a unified framework for multitask adaptation using task-specific LoRA modules, structured prompts, and either explicit routing or merging strategies. This section outlines the prompt design that underpins both training and inference, the task-specific LoRA training procedure, the Mixture-of-LoRAs (MoA) architecture with its router, and a suite of routing-free merging baselines.

3.1 Prompt Design for Task Conditioning

A central design decision in our framework is to leverage highly structured, task-specific prompts that encode task identity directly into the input. These prompts serve two key purposes: (1) they provide the model with explicit cues for task inference, potentially obviating the need for routing; and (2) they enable a consistent data interface for both LoRA module training and router classification.

The prompt formats for each task are, for example in QA:

```
Context: {context}
Question: {question}
Answer:
```

For other tasks, refer to Appendix B.

These prompts are used consistently for both LoRA fine-tuning (paired with targets) and router training (targets discarded). The structured nature of the prompts is critical to our hypothesis that input cues alone may allow for task inference in merged models.

3.2 Task-Specific LoRA Training

Each task T_i is assigned a distinct LoRA module M_i, inserted into the self-attention projection layers of a shared base language model. Following the standard LoRA formulation (Hu et al., 2022), a trainable low-rank update is added to the frozen base weights:

$$W = W_0 + \Delta W_i = W_0 + A_i B_i \cdot \frac{\alpha}{r_i}$$

where $A_i \in R^{d\times r}$, $B_i \in R^{r\times k}$, and $r_i \ll \min(d, k)$ is the LoRA rank. Scaling is done by α. Only A_i and B_i are trained; the base model weights W_0 remain unchanged.

Each LoRA module is trained independently using the prompt-target pairs described above, with standard cross-entropy loss over the target text. No multitask mixing or shared optimization is performed.

3.3 Mixture-of-LoRAs (MoA)

To integrate multiple LoRA modules at inference time, the Mixture-of-LoRAs (MoA) framework introduces a router that selects which task-specific module to activate. This routing is performed at the *sequence level*, once per input, rather than at the token level as in traditional Mixture-of-Experts models.

Router Training. After all LoRA modules have been trained, we construct a classification dataset consisting solely of prompt texts, each labeled with its corresponding task ID. The router module is attached before each transformer block and trained using a *teacher-forcing* approach: all the previous layer's routers selects correct LoRA expert, so that all the routers can be trained in ground-truth hidden states.

The classification head at layer ℓ receives the input hidden state h^ℓ and produces a softmax distribution over tasks. The loss function is the sum of cross-entropy losses across all layers:

$$\mathcal{L}_{\text{router}} = \sum_{\ell=1}^{L} \text{CE}(\text{Router}_\ell(h^\ell), y_{\text{task}})$$

Inference. At inference time, the router selects one task-specific LoRA module to activate. Selection is based on the last token's representation.

3.4 Routing-Free Merging Strategies

We implement and evaluate three distinct strategies for merging LoRA modules trained on different tasks, each corresponding to a different hypothesis about the model's capacity to handle task interference and prompt conditioning. All strategies modify the weight update term ΔW applied on top of a frozen base model.

1. Concatenation-Based Aggregation Instead of selecting a single LoRA per input (as in MoA), this method adds the outputs of all LoRA modules in parallel at each target layer. Given n trained LoRA modules with weights $\{(A_i, B_i)\}_{i=1}^{n}$, each LoRA module computes an update $x \mapsto xA_iB_i$, which is added to the frozen base output:

$$\Delta W_{\text{total}}(x) = \sum_{i=1}^{n} xA_iB_i \cdot \frac{\alpha}{r_i}$$

This is implemented via a custom module `ConcatLoRALinear`, which wraps a base `nn.Linear` and dynamically injects multiple LoRA terms during forward propagation. No routing or gating mechanism is used—task relevance is left to the model to infer from prompt structure alone.

2. Weight Summing In this strategy, all LoRA weights (A_i, B_i) are combined linearly with scalar weights w_i:

$$A_{\text{merged}} = \sum_i w_i A_i, \quad B_{\text{merged}} = \sum_i w_i B_i$$

This produces a single LoRA update that approximates a weighted interpolation of the constituent modules. Weights can be uniform or task-informed. The merged weights are then injected into a standard LoRA module (`LoRALinear`). This approach assumes some degree of parameter compatibility and alignment across LoRA modules.

SVD-Based Merge We implement a rank-reducing merge strategy based on singular value decomposition (SVD). Specifically, we first compute the average of the composed LoRA updates across tasks:

$$\bar{W} = \frac{1}{N} \sum_{i=1}^{N} A_iB_i$$

We then apply SVD to the averaged update matrix:

$$\bar{W} = U\Sigma V^\top$$

To obtain a low-rank approximation of the update, we truncate the top-r singular values and vectors:

$$A_{\text{merged}} = \Sigma_{1:r} V_{1:r}^\top, \quad B_{\text{merged}} = U_{:,1:r}$$

This approach compresses task-specific LoRA updates into a shared low-dimensional subspace, enabling efficient inference with a single merged LoRA module.

3.5 Implementation Details

The router is implemented as a 2-layer MLP attached to the pre-activation hidden state before each transformer block. All prompts and datasets are preprocessed using task-specific formats.

4 Experiments

We evaluate the effectiveness of routing-free LoRA merging strategies compared to the Mixture-of-LoRAs (MoA) framework under a unified multitask setup. The goal of our experiments is to determine whether explicitly selecting task-specific modules is necessary when structured prompts are available to implicitly guide the model.

4.1 Tasks and Datasets

We select five diverse NLP tasks, each representing different input/output formats, linguistic phenomena, and learning challenges. All datasets are standard benchmarks with well-established evaluation protocols:

- **Summarization:** CNN/DailyMail (Hermann et al., 2015), containing approximately 287K news articles and corresponding human-written summaries.
- **Translation:** WMT14 English-German (Bojar et al., 2014), comprising around 4.5 million sentence pairs.
- **Question Answering:** SQuAD v1.1 (Rajpurkar et al., 2016), with over 100K context-question-answer triples based on Wikipedia passages.
- **Sentiment Analysis:** SST-2 (Socher et al., 2013), a binary sentiment classification dataset of short sentences from movie reviews.
- **Natural Language Inference:** MNLI (Williams et al., 2018), a large-scale entailment classification dataset with three labels: entailment, neutral, contradiction.

Additionally, in 7-task setting, we add two more tasks:

- **Paraphrasing:** PAWS (Zhang et al., 2019), a dataset of 108K sentence pairs with high lexical overlap, balanced between paraphrase and non-paraphrase.
- **Commonsense Reasoning:** PIQA (Bisk et al., 2020), a multiple-choice dataset with 17K questions targeting physical commonsense reasoning.

Each dataset is preprocessed into a structured prompt format as described in Section 3. These prompts are used consistently for LoRA training, MoA router training, and evaluation.

4.2 Model Variants

We evaluate the following five configurations:

1. **Base:** The unmodified Qwen-2 0.5B model without any task-specific adaptation. Serves as a zero-shot baseline.
2. **MoA (Mixture-of-LoRAs):** A separate LoRA is trained per task, and a learned router dynamically selects the appropriate module at inference time.
3. **Concatenation:** All LoRA modules are applied in parallel; their outputs are summed without any selection.
4. **weight summing:** A fixed-weight linear combination of all LoRA modules is computed and used universally.
5. **SVD Merge:** LoRA modules are averaged and compressed into a single low-rank module via SVD.

Each model is evaluated on all five tasks in a uniform setting. Merged models do not rely on any routing or external task metadata.

4.3 Evaluation Metrics

We use standard, task-specific metrics for comparison:

- **Summarization:** ROUGE-1, ROUGE-2, ROUGE-L (Lin, 2004).
- **Translation / Paraphrasing:** BLEU (Papineni et al., 2002).

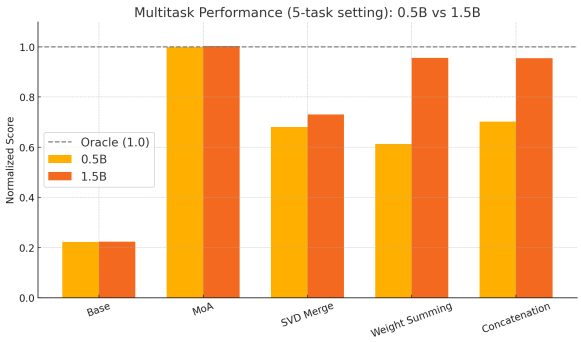

Figure 1: Multitask ability scores for 0.5B and 1.5B models in 5-task setting. Normalized by individual task-specific oracle performance.

- **Question Answering:** Exact Match (EM) and F1, using official SQuAD evaluation scripts.
- **Sentiment Analysis / NLI / Commonsense Reasoning:** Classification accuracy.

All scores are computed using evaluation subsets held out from the training set. No task-specific hyperparameter tuning is performed to ensure a fair comparison across models.

4.4 Training Configuration

All models use the Qwen-2 decoder-only architecture as the base. LoRA modules are inserted into all attention projection layers with a rank of 8.

For MoA, router training is performed after all LoRA modules are trained. The router receives the prompt-only input (no target) and is trained using cross-entropy loss with task labels. Accuracy of router predictions is measured on a held-out validation set, and averaged across all transformer layers.

All merged models are created post-hoc without additional fine-tuning. For SVD merging, we use top-8 rank approximation to match the original LoRA rank. weight summings use uniform coefficients unless otherwise stated.

5 Results

Table 1, and Table 2 each summarizes the performance of Qwen 0.5B and 1.5B models in 5-task settings. Table 3 shows overall multitask scores calculated from it. Table 4 shows overall multitask scores of 1.5B model, in 7-task settings.

Model	Summarization (ROUGE)			Translation	QA (SQuAD)		Sentiment Acc.	NLI Acc.
	R-1	R-2	R-L	BLEU	EM	F1		
Base	0.143	0.041	0.108	0.006	25.6	36.0	0.000	0.000
MoA	**0.267**	0.092	**0.194**	**0.093**	**67.4**	**71.4**	**0.904**	**0.800**
SVD Merge	0.167	0.051	0.127	0.074	48.6	55.5	0.836	0.198
weight summing	0.233	0.083	0.177	0.061	61.2	69.5	0.200	0.251
Concatenation	0.254	**0.093**	0.191	0.074	64.2	70.2	0.318	0.331
Individual LoRA (Oracle)	0.269	0.095	0.197	0.093	67.6	71.7	0.898	0.788

Table 1: Performance comparison across model variants using a 0.5B model, in 5-task setting. The final row shows individual task-specific LoRA modules, serving as an oracle upper bound. Best results per column (excluding oracle) are in bold.

Model	Summarization (ROUGE)			Translation	QA (SQuAD)		Sentiment Acc.	NLI Acc.
	R-1	R-2	R-L	BLEU	EM	F1		
Base	0.160	0.046	0.119	0.016	15.4	34.9	0.004	0.000
MoA	0.276	0.100	0.201	**0.148**	78.8	82.6	0.930	**0.871**
SVD Merge	0.156	0.046	0.117	0.100	70.0	77.4	0.912	0.400
weight summing	0.276	0.102	0.206	0.121	**79.8**	83.3	0.928	0.796
Concatenation	**0.278**	**0.103**	**0.210**	0.119	78.8	**83.6**	**0.932**	0.780
Individual LoRA (Oracle)	0.278	0.101	0.203	0.144	78.6	82.7	0.926	0.874

Table 2: Performance comparison across model variants using a 1.5B model, in 5-task setting. The final row shows individual task-specific LoRA modules, serving as an oracle upper bound. Best results per column (excluding oracle) are in bold.

Model	0.5B	1.5B
Base	0.223	0.224
MoA	0.999	1.004
SVD Merge	0.681	0.731
weight summing	0.613	0.956
Concatenation	0.702	0.955
Individual LoRA (Oracle)	1.000	1.000

Table 3: Relative multitask ability scores for 0.5B and 1.5B models, in **5-task setting**. Each score is the average performance normalized to task-specific oracle values.

Model	1.5B
Base	0.242
MoA	1.002
SVD Merge	0.704
weight summing	0.790
Concatenation	0.771
Individual LoRA (Oracle)	1.000

Table 4: Multitask ability scores for 1.5B models, in **7-task setting**. Scores are computed as the average of task-specific representative metrics normalized to oracle values.

6 Analysis and Discussion

6.1 MoA: Near-Oracle Performance via Accurate Task Routing

The Mixture-of-LoRAs (MoA) architecture achieves performance that is consistently close to the oracle setting in which each task-specific LoRA is applied individually. As shown in Table 1 and Table 2, the MoA model delivers nearly identical results to the *Individual LoRA (Oracle)* across all tasks. This indicates that the router is highly effective at classifying tasks from structured input prompts and consequently applying the correct LoRA module at inference.

Notably, this modular design allows for clean separation between task representations, avoiding the interference typically seen in multitask adaptation. The closeness to oracle performance confirms that explicit routing is not only effective but also sufficient for preserving task-specific performance without degrading generalization.

6.2 Router-Free Merging Improves with Model Scale

While router-based methods like MoA dominate at smaller scales (e.g., 0.5B), the gap between MoA and router-free approaches narrows significantly as model capacity increases. In the 1.5B setting, simple strategies such as weight summing and even naive concatenation match or exceed MoA in certain metrics:

- **Concatenation** surpasses MoA on summa-

Method	Summarization (ROUGE)			Translation	QA (SQuAD)		Sentiment Acc.	NLI Acc.
	R-1	R-2	R-L	BLEU	EM	F1		
Concatenation	**0.278**	**0.103**	**0.210**	**0.119**	**78.8**	**83.6**	**0.932**	**0.780**
w/o Scaling	0.225	0.077	0.176	0.092	75.8	81.2	0.888	0.578
weight summing	**0.276**	**0.102**	**0.206**	**0.121**	**79.8**	**83.3**	**0.928**	**0.796**
Averaged	0.222	0.080	0.174	0.088	76.2	81.8	0.920	0.708

Table 5: Effect of LoRA scaling factor and weighting scheme in router-free merging at 1.5B scale. Removing scaling (for concatenation) or using uniform weights (for weight summing) causes a significant drop in performance across all tasks.

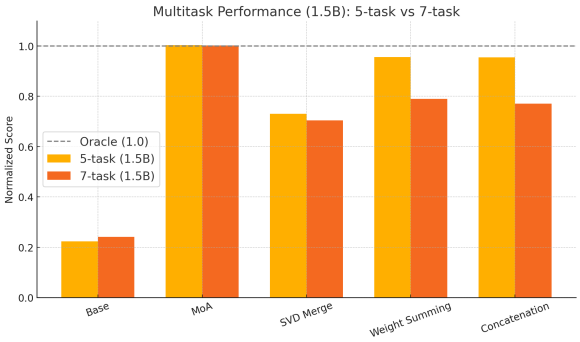

Figure 2: Multitask ability scores for 1.5B models in 5-task and 7-task setting. Normalized by individual task-specific oracle performance.

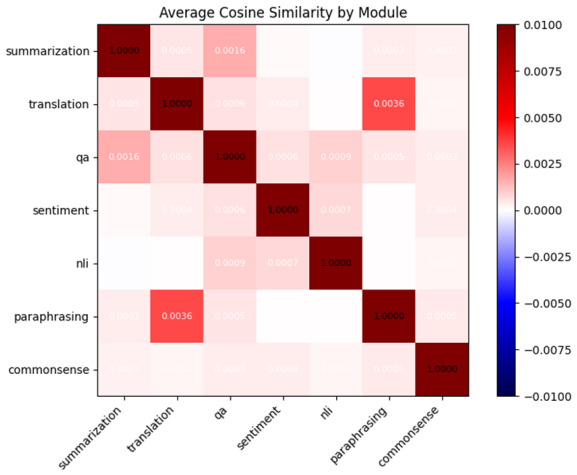

Figure 3: The average cosine similarities of LoRA weights in each tasks.

rization (ROUGE) and QA (F1), indicating emergent task specialization without explicit routing.

- **weight summing**, previously underperforming, now matches MoA in QA (EM) and performs the best overall, in router-free methods.

These trends are corroborated by the normalized multitask ability scores in Table 3, where both concatenation and weight summing nearly reach the oracle level (0.955 and 0.956 respectively, versus 1.000).

This suggests that larger models inherently develop stronger context sensitivity and can exploit structured prompts to internally route to task-appropriate behavior—even in the absence of architectural support. In other words, *scaling enables implicit routing*. Check Figure 1 for overall comparison.

6.3 Orthogonality of the LoRA weights

In Figure 3, it shows the average cosine similarities of LoRA weights. We can see that the cosine similarity is almost zero in every pairs when the tasks are different, meaning extreme orthogonality across the tasks. Because of this, the language model might be able to use appropriate weights without interference, in methods that linearly combine the weights.

6.4 Ablation 1: Preserving LoRA weights and scale is Crucial

Table 5 shows that using the default concatenation method, which retains each LoRA module's scaling and applies them in parallel during inference, consistently outperforms its ablated version where scaling is not applied. For example, translation BLEU drops from 0.119 to 0.092, and NLI accuracy decreases from 0.780 to 0.578.

A similar pattern is observed in the weight summing strategy: naively averaging LoRA weights significantly degrades performance compared to simply summing them. The averaged variant sees a drop of performance in all tasks.

Therefore, router-free methods work best, when not aggressively compressing or modifying weights, but when respecting the integrity of both the model's computation graph and the individual LoRA representations.

6.5 Ablation 2: More Tasks make Performance Degradation in Router-Free Methods

As observed by comparing the 1.5B model results in Table 3 and Table 4, the performance of all router-free methods consistently degrades as the number of tasks increases. In contrast, the MoA model maintains robust performance across tasks. This suggests that naive merging of LoRA weights leads to interference between tasks, limiting the scalability of router-free approaches in multi-task settings as the number of tasks increase. We can easily check this result in Figure 2, and the full results are in Appendix A.

7 Conclusion

In this work, we explored whether routing mechanisms are necessary for effective multitask adaptation in large language models when using task-specific LoRA modules. While the Mixture-of-LoRAs (MoA) framework explicitly routes inputs to the appropriate module using a learned classifier, we proposed and evaluated routing-free alternatives that merge LoRA modules via simple strategies such as concatenation, weight summing, and SVD compression.

Our experiments across diverse NLP tasks show that:

- MoA achieves strong performance in all cases, as the structured tasks are good target for router classification.
- Concatenation, and weight summing gives surprisingly good results, especially in large models, with fewer tasks.
- SVD merging offers a balanced alternative that compresses multiple task-specific modules into a single LoRA while maintaining robust performance, even if the task increases.

These findings suggest that, for tasks with clearly structured prompts and minimal semantic overlap, routing may not be essential. Instead, lightweight merging strategies can offer comparable performance with significantly reduced architectural complexity and no runtime task classification.

Limitations Due to lack of GPU resources, we were only able to do some experiments on small models, maximum 1.5B model. But as we see, 0.5B models also show some consistent results, and we believe that the effectiveness of given methods will also be seen in some bigger size language models.

References

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Pavel Pecina, Matt Post, Daniel Zeman, and 1 others. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*.

Tianyu Feng, Wenhui Sun, Jie Chen, Jie Yang, Shuaicheng Wang, Yichong Wu, and Yelong Chen. 2024. Mixture-of-loras: An efficient multitask tuning method for large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics and Language Resources (LREC-COLING)*.

Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2014. An empirical investigation of catastrophic forgetting in gradient-based neural networks. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*.

Karl Moritz Hermann, Tom Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NeurIPS)*.

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 (ICML)*, pages 2790–2799. PMLR.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *Proceedings of the 10th International Conference on Learning Representations (ICLR)*.

Gabriel Ilharco, Mitchell Wortsman, Cade Gordon, Luke Zettlemoyer, Hannaneh Hajishirzi, Ali Farhadi, and Pang Wei Koh. 2023. Editing models with task arithmetic. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. Gshard: Scaling giant models with conditional computation and automatic sharding. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jonas Pfeiffer, Akhilesh Kamath, Sebastian Ruder, and Kyunghyun Cho. 2020. Adapterfusion: Non-destructive task composition for transfer learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Victor Sanh, Albert Webson, Colin Raffel, Shaden Bach, Lintang Sutawika, Zaid Alyafeai, Alex Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, and 1 others. 2022. Multitask prompted training enables zero-shot task generalization. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. In *Proceedings of the 10th International Conference on Learning Representations (ICLR)*.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Mitchell Wortsman, Gabriel Ilharco, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Shibani Dave, Vaishaal Shankar, Benjamin Recht, Rebecca Roelofs, and 1 others. 2022. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. PAWS: Paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, Minneapolis, Minnesota. Association for Computational Linguistics.

A 7-task results

We report detailed 7-task results in Qwen2-1.5B model. See Table 6. As we can see, the router-free methods show lower performance compared to the 5-task setting, and MoA still remains strong, near the oracle performance.

B Task Formats

We report the formats used for all 7 tasks, used in the experiment.

Prompt Format – Summarization

```
{Article}
### Summarization:
```

Prompt Format – Translation

```
{english}
### Translation from English to German:
```

Table 6: Full 7-task evaluation results (1.5B model). Part 1: summarization, translation, QA, sentiment, and NLI.

Model	**Summarization (ROUGE)**			**Translation (BLEU)**	**QA (SQuAD)**		**Sentiment Acc.**	**NLI Acc.**
	R-1	R-2	R-L		EM	F1		
Base	0.161	0.047	0.119	0.016	15.4	34.9	0.004	0.000
MoA	**0.276**	**0.099**	**0.201**	**0.145**	**79.2**	**83.0**	**0.930**	**0.865**
SVD Merge	0.159	0.046	0.118	0.097	62.6	71.6	0.868	0.053
weight summing	0.151	0.064	0.124	0.103	72.0	77.0	0.866	0.647
Concatenation	0.194	0.077	0.157	0.105	72.6	78.2	0.896	0.578
Individual LoRA (Oracle)	0.278	0.100	0.203	0.139	79.4	83.1	0.930	0.867

Table 6: (continued) Part 2: paraphrase generation and commonsense reasoning.

Model	**Paraphrase (BLEU)**	**Commonsense Acc.**
Base	0.109	0.280
MoA	**0.572**	**0.734**
SVD Merge	0.553	0.616
weight summing	0.508	0.510
Concatenation	0.511	0.300
Individual LoRA (Oracle)	0.573	0.746

Prompt Format – Question Answering

Context:
{context}
Question:
{question}
Answer:

Prompt Format – Commonsense Reasoning

{goal}
Solution 1:
{sol1}
Solution 2:
{sol2}
Correct Solution:

Prompt Format – Sentiment Analysis

{text}
Sentiment:

Prompt Format – Natural Language Inference (NLI)

Premise:
{premise}
Hypothesis:
{hypothesis}
Relationship:

Prompt Format – Paraphrasing

{sentence}
Paraphrasing: