LoraRetriever: Input-Aware LoRA Retrieval and Composition for Mixed Tasks in the Wild

Ziyu Zhao[1], Leilei Gan[2]*, Guoyin Wang[3], Wangchunshu Zhou[4], Hongxia Yang[3], Kun Kuang[1,5]*, Fei Wu[1,6,7]*

[1]Department of Computer Science and Technology, Zhejiang University,
[2]College of Software and Technology, Zhejiang University, [3]ByteDance Inc.,
[4]AIWaves Inc., [5]Law&AI Lab, Zhejiang University, [6]Shanghai AI Laboratory
[7]Shanghai Institute for Advanced Study, Zhejiang University
{ziyuzhao.cs, leileigan, kunkuang, wufei}@zju.edu.cn

Abstract

Low-Rank Adaptation (LoRA) provides an effective yet efficient solution for fine-tuning large language models (LLMs). The modular and plug-and-play nature of LoRA enables the integration of diverse domain-specific LoRAs to enhance the capabilities of LLMs. Previous research on exploiting multiple LoRAs either focuses on specific isolated downstream tasks or fixes the selection of LoRAs during training. However, in real-world scenarios, LLMs receive diverse prompts covering different tasks, and the pool of candidate LoRAs is often dynamically updated. To bridge this gap, we propose LoraRetriever, a retrieve-then-compose framework that adaptively retrieves and composes multiple LoRAs according to the input prompts. LoraRetriever contains three main components: firstly, identifying and retrieving LoRAs relevant to the given input; secondly, formulating strategies for effectively integrating the retrieved LoRAs; and thirdly, developing efficient batch inference to accommodate heterogeneous requests. Experimental results indicate that LoraRetriever consistently outperforms the baselines, highlighting its practical effectiveness and versatility. Our code is available at https://github.com/StyxXuan/LoraRetriever.

1 Introduction

Recently, large language models (LLMs) such as ChatGPT (Liu et al., 2023b) and Llama (Touvron et al., 2023) have shown notable successes in a range of fields (Hadi et al., 2023; Wang et al., 2023; Wu et al., 2023). Nevertheless, due to the prohibitively high computation costs for fine-tuning LLMs on specific domains, there is a growing shift towards Parameter-Efficient Fine-Tuning (PEFT) (Liu et al., 2022; Hu et al., 2023, 2021; Zhao et al., 2024), which only updates a small fraction of the model's parameters or integrates new trainable parameters that augment the model's capabilities. Within this sphere, Low-Rank Adaptation (LoRA) (Hu et al., 2021) stands out for its remarkable effectiveness, modularity, and plug-and-play capabilities. As AI communities like Huggingface and ModelScope witness an influx of LoRA parameters tailored for diverse tasks and domains, there is an increasing emphasis on employing various LoRA experts to provide a comprehensive service (Sheng et al., 2023).

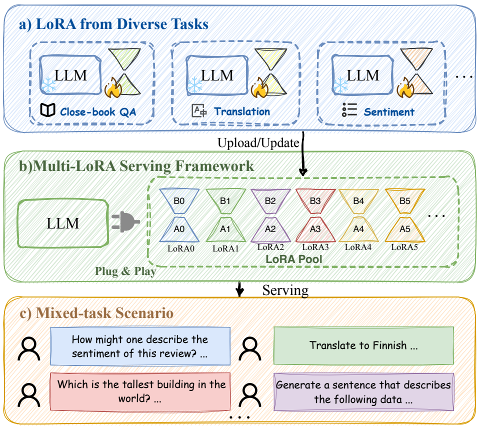

a) LoRA from Diverse Tasks
LLM
LLM
LLM
Close-book QA
Translation
Sentiment
...
Upload/Update
b)Multi-LoRA Serving Framework
LLM
B0
B1
B2
B3
B4
B5
A0
A1
A2
A3
A4
A5
LoRA0
LoRA1
LoRA2
LoRA3
LoRA4
LoRA5
...
LoRA Pool
Plug & Play
Serving
c) Mixed-task Scenario
How might one describe the sentiment of this review? ...
Translate to Finnish ...
Which is the tallest building in the world? ...
Generate a sentence that describes the following data ...
...

Figure 1: Illustration of serving multiple LoRAs within a dynamically updated LoRA pool for mixed-task scenarios. a) LoRAs from various domains and tasks aimed at enhancing specific capabilities of the LLM can be uploaded to or updated to the LoRA pool. b) The multi-LoRA serving framework aims to leverage the plug-and-play nature of LoRAs to offer comprehensive services. c) The downstream tasks, presented in a mixed-task form, require personalized expert routing.

Recent research has explored the integration of the mixture of expert (MoE; Jacobs et al. (1991); Jordan and Jacobs (1994)) with LoRAs (Wang et al., 2022; Liu et al., 2023a; Zadouri et al., 2023; Muqeeth et al., 2023; Anonymous, 2024). However, these methods lock in the selection of LoRAs while training, lacking the ability to dynamically update and scale in scenarios where the LoRA pool may consistently expand. LoRAhub (Huang

*Corresponding authors.

August 11-16, 2024 ©2024 Association for Computational Linguistics

et al., 2023) and AdapterSoup (Chronopoulou et al., 2023) explore composing LoRAs for specific downstream tasks. However, these two methods offer a one-size-fits-all solution for downstream tasks, overlooking the heterogeneous nature of diverse real-world requests.

To bridge these gaps, our paper explores the "mixed-task scenario" as exemplified by platforms like ChatGPT (Liu et al., 2023b) and Gemini (Team et al., 2023), wherein the nature of user requests encompasses diverse prompts covering different tasks. While LLMs present a unified solution for a broad spectrum of tasks, their performance can still falter in certain specialized areas(Liu et al., 2023a; Yang et al., 2023; Chai et al., 2024). This is where the integration of LoRAs becomes crucial. As shown in Fig.1, our vision encompasses a multi-LoRA serving framework capable of dynamic enhancement, continuously improving its functionality as new LoRA modules are added and updated. Through the plug-and-play capabilities of LoRA, the framework can provide personalized services for heterogeneous downstream requests.

In this paper, we introduce LoraRetriever, a retrieve-then-compose framework designed to exploit the plug-and-play nature of LoRA in mixed-task scenarios. Our framework consists of three key components: (1) **Input-aware LoRA Retrieval:** The first step of our framework is aligning user inputs with the corresponding LoRAs through sentence embeddings and is further refined by an instruction fine-tuning (Su et al., 2022; Asai et al., 2022) for effective LoRA retrieval. Through the retriever, we achieve a more flexible LoRA routing mechanism, whose training stage is disentangled from the training and inference of the LLM. (2) **LoRA Composition:** Our framework next employs two strategies for compositing the retrieved LoRAs in the first step. The *Fusion of LoRAs* averages multiple LoRAs' parameters and constructs a singular comprehensive model for each input. The *Mixture of LoRAs* activates multiple LoRAs simultaneously and then averages the output of each submodule of the LoRAs. Compositing top-k LoRAs increases the recall rate for the correct LoRA and improves the generalization of unseen tasks by integrating the LoRAs of similar tasks. (3) **Batch Inference of Multiple LoRAs:** Most previous work on the input-adaptive inference of LLMs does not support batch inference (Zhou et al., 2020; Chronopoulou et al., 2023). To tackle the challenge of heterogeneous batched requests, we construct a unique LoRA mapping matrix for batch samples. This allows for tailored inferences through efficient matrix multiplication, ensuring each request activates its corresponding LoRAs while maintaining batch processing efficiency.

To assess the performance of LoraRetriever, we established a mixed-task evaluation benchmark comprising 48 LoRAs spanning a variety of natural language understanding and generation tasks. The experimental results underline the effectiveness of the proposed methods in serving both in-domain and out-of-domain downstream requests. Furthermore, the retrieval routing method exhibits a robust generalization capability: although the retriever is trained on just 40% of the tasks, it effectively retrieves the corresponding LoRAs for unseen tasks.

2 Related Work

Mixture of Experts. The Mixture of Experts (MoE) method combines various specialized submodules, guided by a gating network to tailor responses to different input types (Jacobs et al., 1991; Jordan and Jacobs, 1994; Shen et al., 2023; Riquelme et al., 2021; Dou et al., 2023). Some work (Wang et al., 2022; Zadouri et al., 2023; Zhu et al., 2023; Liu et al., 2023a; Dou et al., 2023) focuses on using the MoE method for PEFT to achieve more effective and efficient model fine-tuning. Other work (Anonymous, 2024; Muqeeth et al., 2023) focuses on using MoE to coordinate existing LoRA experts without specifically training the experts' parameters. These methods require training additional parameters for gating and are limited to a fixed number of LoRAs, making them unsuitable for complex and dynamic scenarios. Our method can be seen as gating through a retriever, hence achieving flexibility and generalization on unseen experts.

Adapter Merging. In addition to model ensembling through the MoE, there is an increasing focus on aggregating adapters from different domains through the method of Adapter Merging. AdapterSoup (Chronopoulou et al., 2023) aggregates different adapters in the parameter space, allowing large language models to adapt to new domains without additional training. LoRAhub (Huang et al., 2023) employs random sampling of LoRA parameters from various domains and tasks, followed by black-box optimization to learn the weights of different LoRA parameters without involving model gradient backpropagation. These methods offer a one-size-fits-all solution for downstream tasks, which cannot be applied in the mixed-task scenario for providing personalized service.

3 Preliminaries

This section begins with a concise introduction to the Low-Rank Adaptation, followed by a detailed formalization of the mixed-task scenario.

3.1 Low-Rank Adaptation

Directly fine-tuning large language models with all parameters is computationally intensive and is not feasible in low-resource scenarios. Based on the idea that only a small number of low-rank parameters need to be fine-tuned for sufficient performance in new domains, Hu et al. (2021) proposed the Low-Rank Adaptation, where the LoRA module can be combined with the pre-trained parameters in parallel for efficient inference.

Specifically, given pre-trained weights $W_0 \in \mathbb{R}^{d\times d}$ of a sub-module of LLM, the LoRA adds an extra trainable weight matrix as $W_0 + \Delta W = W_0 + BA$, where ΔW can be decomposed into two smaller matrices $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times d}$, where r stands for the rank of ΔW. The forward pass can be modified as follows:

$$x' = W_0 x + \Delta W x = W_0 x + BAx, \qquad (1)$$

where $x \in \mathbb{R}^d$ is the input and the $x' \in \mathbb{R}^d$ denote the output.

3.2 Problem Formulation

In this part, we give a formal definition of the mixed-task scenario. Given an original LLM L, we have a set of k LoRAs, $\Phi = \{\phi_1, \phi_2, \cdots, \phi_k\}$, on the shelf, where each LoRA ϕ_i is trained on its corresponding task T_i. The mixed task inputs can be formulated as $T_{mix} = \{x, \forall x \in T_1 \vee T_2 \cdots \vee T_k\}$, where $\vee$ stands for the logical disjunction operator.

Under the mixed-task scenario, given an input $x \in T_{mix}$ without its task tag, the serving process can be written as:

$$y = F(g(\Phi, x), x, \theta), \qquad (2)$$

where θ denotes the original parameters of LLM, $g(\Phi, x)$ represents the input-aware LoRA retrieval process and returns a set of retrieved LoRAs Φ_i. $F(\Phi_i, x_i, \theta)$ depicts the LoRA composition process that integrates the retrieved LoRAs as a plug-in to the original LLM.

4 LoraRetriever Framework

In this section, we describe the LoraRetriever framework as shown in Fig.2 for serving multi-LoRAs in mixed-task scenarios. This framework contains three major components: the input-aware LoRA retrieval module (§4.1), the LoRA composition module (§4.2), and the batch inference strategy (§4.3).

4.1 Input-Aware LoRA Retrieval

Our goal is to construct a LoraRetriever tailored to effectively retrieve the corresponding LoRAs for each input in scenarios where LoRAs are dynamically updated. However, existing approaches fall short of accurately identifying LoRAs under such conditions. MoE-based methods (Anonymous, 2024; Muqeeth et al., 2023) struggle to generalize when new LoRAs are introduced due to the fixed selection of LoRAs established during router training. Retrieval methods like sentence embedding (Reimers and Gurevych, 2019; Ni et al., 2021) or task embedding (Achille et al., 2019; Zhou et al., 2022) fail to map both samples and LoRA into a shared embedding space, limiting their effectiveness in input-aware LoRA retrieval.

To achieve this goal, we propose to train a retriever via instruction fine-tuning (Su et al., 2022; Asai et al., 2022), namely LoraRetriever, which can retrieve suitable LoRAs from a massive LoRA pool for a given input sample. The fundamental concept behind LoraRetriever comprises two main steps: (i) First, to embed different task-specific LoRAs into embedding space for facilitating retrieval, we posit that each LoRA can be represented by some data points, which can be obtained by randomly choosing a dozen samples from the training dataset. Then we average their instruction embeddings to represent the embedding of each LoRA. (ii) To improve generalization for unseen LoRAs in LoRA retrieving, we train the retriever through instruction fine-tuning (Su et al., 2022; Wei et al., 2021) on a subset of all tasks. Training on a small subset of tasks is designed to simulate scenarios involving the integration of new LoRAs, thereby underscoring our method's generalization abilities via instruction fine-tuning. These two strategies enable the effective use of limited data distributions for input-aware retrieval and can be generalized to unseen LoRAs.

Formally, with a sentence-embedding model E, input sequence x, and the instruction I for

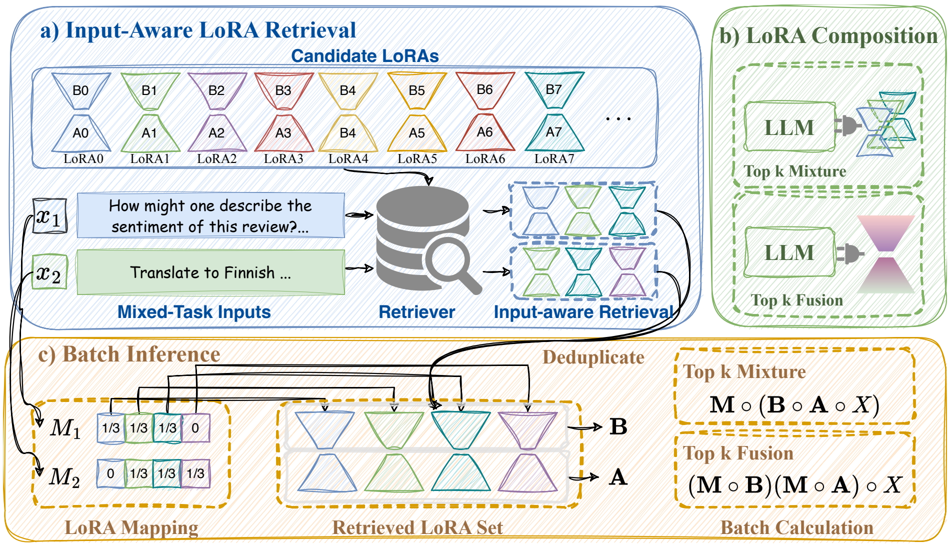


Figure 2: The LoraRetriever Framework. This framework, equipped with a pool of candidate LoRAs from various domains/tasks, is designed to offer personalized services tailored to the input provided. It begins by executing an input-aware LoRA retrieval process aimed at identifying LoRAs corresponding to tasks analogous to the input (§4.1). Subsequently, it employs a specialized LoRA composition mechanism to efficiently utilize the retrieved LoRAs (§4.2). By constructing a LoRA mapping matrix for batch inputs, the framework facilitates effective batch inference (§4.3).

embedding purposes, the instructed embedding can be formulated as $E(I \oplus x)$, where $\oplus$ denotes the concatenation operation. In order to allow the embedding to capture the similarity between different tasks, the instruction is expressed as: *"Represent the sentence for similar task retrieval"*. Each LoRA module is embedded with m randomly selected domain-specific samples, expressed as $E(\phi) = \frac{1}{m}\sum_{i=1}^{m} E(I \oplus x_{i\phi})$. This embedding method integrates both sample-wise and LoRA-module-wise embeddings, facilitating the calculation of similarity between an individual sample and a LoRA module. For measuring the similarity between LoRA module ϕ and the input sequence x, following (Ni et al., 2021), we leverage the cosine similarity between the LoraRetriever embeddings: $s(x, \phi, I) = \cos(E(I \oplus x), E(\phi))$.

To improve LoRA retrieval by the retriever and broaden its generalization to unseen LoRAs, we train the embedding model E through instruction fine-tuning on a small subset of tasks. To prevent the need to access new samples, we use previously employed samples for embedding LoRAs as our training data. Consider t distinct training tasks, represented as $\mathcal{T}_{train} = \{T_1, \cdots, T_t\}$. Following Ni et al. (2021), the training dataset $\mathcal{D}$ comprises paired samples (x_i, x_i^+), where each x_i is a sample from a task $T_i \in \mathcal{T}_{train}$, and a positive sample x_i^+ is randomly selected from the same task T_i. To complement each positive pair, we randomly select p negative pairs $(x_i, x_{ij}^-)_{j=1}^{p}$, ensuring that x_{ij}^- is sourced from tasks outside of T_i, thereby $x_{ij}^- \notin T_i$. The training process is achieved through a contrastive loss (Karpukhin et al., 2020; Izacard et al., 2021; Ni et al., 2021) defined as follows:

$$\mathcal{L} = \frac{e^{s(x_i, x_i^+, I)/\gamma}}{e^{s(x_i, x_i^+, I)/\gamma} + \sum_{j=1}^{p} e^{s(x_i, x_{ij}^-, I)/\gamma}},$$

where γ is the softmax temperature.

During the LoRA retrieval phase, the top-k LoRAs are retrieved according to their similarity to the input x. This process can be formulated as follows:

$$g(x_i, \Phi) := \Phi_i = \mathrm{TopK}\{s(\phi_j, x_i, I), \phi_j \in \Phi\}.$$

4.2 LoRA Composition

After retrieving the top-k LoRAs Φ_i for an input x_i, we proceed to integrate these LoRAs into the LLM with parameter θ. This integration is achieved by applying two different LoRA composition strategies: the *Mixture of LoRAs* and the *Fusion of LoRAs*.

4.2.1 Mixture of LoRAs

The mixture of LoRAs strategy involves the aggregation of the outputs of each submodule within the assembled LoRAs. Let us denote $\mathbf{A} = \{A_1, A_2, \ldots, A_n\}$ and $\mathbf{B} = \{B_1, B_2, \ldots, B_n\}$

as the sets representing submodules within n LoRAs. For an input x_i, the output derived from the mixture of LoRAs can be expressed as $x'_i = \frac{1}{n}\sum_{j=1}^{n} B_j A_j x_i$, where x'_i denotes the output. This process signifies the integration of each LoRA module's output, effectively blending their contributions to form a unified output.

4.2.2 Fusion of LoRAs

In contrast to the Mixture method, which combines the output of different LoRAs, fusing the parameters of these LoRAs presents an alternative composition strategy.

Let the parameters of each LoRA ϕ_i be denoted by Θ_i. The parameter of the fused LoRA is then represented as $\Theta_{\text{fusion}} = \frac{1}{k}\sum_{j=1}^{k} \Theta_j$. This formulation allows the fused parameter to function akin to a single LoRA.

4.3 Batch Inference of Multiple LoRAs

Implementing batch inference in the presence of multiple LoRAs and diverse composition diagrams poses a significant technical challenge. To address this, we introduce a unique approach for batch inference. Our method involves processing a batch of samples denoted as $X \in \mathbb{R}^{b \times l \times d}$, where b, l, and d denote the batch size, sequence length, and sample dimensionality, respectively. For each input x_i and its retrieved LoRAs Φ_i within the same batch, we aggregate these LoRAs into a collective set denoted by $\Phi_{\mathcal{B}}$. To ensure the uniqueness of $\Phi_{\mathcal{B}}$, we eliminate duplicates, mindful of the possibility that retrieved LoRAs may overlap across different samples. The resulting set $\Phi_{\mathcal{B}}$ comprises p unique LoRAs, where $p \leq bk$. For every sample x_i, a p dimension mapping vector M_i is generated, which specifies the indices of its corresponding LoRAs within $\Phi_{\mathcal{B}}$.

The LoRA mapping vectors are combined into a matrix $\mathbf{M} \in \mathbb{R}^{b \times p}$. The parameters of a submodule in LoRA can be denoted as A and B, and are concatenated within the batched LoRAs $\Phi_{\mathcal{B}}$ to obtain $\mathbf{A} \in \mathbb{R}^{p \times r \times d}$ and $\mathbf{B} \in \mathbb{R}^{p \times d \times r}$. The batch inference process of the mixture of LoRAs can be formulated as follows:

$$X' = \mathbf{M} \circ (\mathbf{B} \circ \mathbf{A} \circ X), \tag{3}$$

where we denote the batched output of a layer of multiple LoRA as $X' \in \mathbb{R}^{b \times l \times d}$ and extend the symbol $\circ$ to denote potential broadcasting as Wen and Chaudhuri (2023). The batch inference process of LoRA fusion can be formulated as

$$X' = (\mathbf{M} \circ \mathbf{B})(\mathbf{M} \circ \mathbf{A}) \circ X. \tag{4}$$

These strategies can be simply implemented by the einsum operation, and the PyTorch-style pseudocode is shown in Appendix.D.

5 Experiments

This section outlines the evaluation framework for assessing different approaches in mixed-task scenarios. Furthermore, a comprehensive analysis of the proposed LoraRetriever framework is presented.

5.1 Evaluation Framework

Base Model & LoRA Configuration. To test various methods in the mixed-task scenarios, we leverage Llama-2-{7b,13b} (Touvron et al., 2023) as the base models and train a range of LoRAs for a spectrum of tasks. We select a portion of the Flan-v2 datasets (Wei et al., 2021) to train 48 LoRAs for a spectrum of tasks covering Natural Language Understanding (NLU) and Natural Language Generation (NLG). Following the categorization by Wei et al. (2021), these tasks can be grouped into 10 distinct task clusters. We train each LoRA according to the Alpaca (Taori et al., 2023) format and rank r, and the scaling hyperparameter α are set to 6 and 12, respectively. The details of the LoRAs training can be found in the Appendix. E.

Mixed Task Evaluation Dataset. For constructing the mixed-task dataset, we randomly chose 50 samples from the test set for each task used in training 48 LoRAs, subsequently mixing and shuffling these samples to form a unified dataset with 6000 data entries. Further details about these datasets are available in the Appendix.E.

Baseline Methods. We compared our method with the following baselines: (1) **Mixture of Experts** (Zhu et al., 2023; Zadouri et al., 2023; Liu et al., 2023a; Wang et al., 2022; Anonymous, 2024). (2) **SMEAR** (Muqeeth et al., 2023); (3) **Adapter-Soup** (Chronopoulou et al., 2023); (4) **LoRAhub** (Huang et al., 2023). Specifically, we implement three variants of MoE. A detailed description of the baseline models can be found in Appendix.A, with their implementations presented in Appendix.F.

Implementation of LoraRetriever. To train the LoraRetriever, we continue to perform instruction

Task	Perfect Selection	Selection		Fusion		Mixture		MoE	MoE	MoE	SME-	Adapter	LoRA
		IID	OOD	IID	OOD	IID	OOD	Top1	Top3	Soft	AR	Soup	Hub
						w/ Llama2-7b							
Struct to Text$_{Rouge-1}$	64.0	**61.3**	50.1	49.4	45.9	55.9	50.4	45.6	46.8	47.9	48.0	4.5	35.6
Struct to Text$_{Rouge-2}$	39.6	**37.0**	26.6	25.7	23.5	30.0	26.4	21.9	22.9	23.8	24.2	1.1	17.7
Struct to Text$_{Rouge-l}$	57.0	**54.5**	43.9	43.6	40.3	49.5	44.0	39.8	40.7	41.7	42.4	4.5	31.6
Translation$_{BLEU}$	13.1	**12.8**	12.0	12.2	12.3	**12.8**	12.2	9.5	10.5	10.7	11.0	1.4	8.5
COMMONSENSE	62.5	55.5	46.0	51.0	48.0	**61.5**	50.0	54.5	52.0	51.5	50.0	46.0	17.5
SENTIMENT	90.0	**89.5**	89.0	79.0	78.5	**89.5**	90.5	70.0	75.0	74.5	74.0	73.5	0.5
READING Comp.	67.3	**51.7**	40.3	47.3	45.0	51.3	47.3	48.7	47.7	48.7	45.7	40.7	2.7
CLOSE-BOOK QA	45.0	40.0	43.0	41.0	37.5	**45.0**	48.5	40.5	38.5	40.0	32.0	31.5	1.0
COREFERENCE	52.0	50.0	46.0	47.0	53.0	**63.0**	49.0	61.0	59.0	57.0	58.0	43.0	1.0
READ. COOMP. W/ COM	69.0	**69.0**	30.0	35.0	19.0	46.0	40.0	31.0	29.0	29.0	23.0	14.0	3.0
PARAPHRASE	65.5	**58.0**	45.5	45.5	44.0	56.5	45.5	42.0	38.5	36.0	34.5	46.5	1.0
NLI	72.3	**70.0**	60.6	51.4	53.8	67.9	64.3	50.3	49.6	48.3	50.8	62.4	10.5
						w/ Llama2-13b							
Struct to Text$_{Rouge-1}$	65.4	**62.6**	49.4	52.7	49.7	57.7	52.1	46.8	47.0	48.5	48.3	7.1	39.3
Struct to Text$_{Rouge-2}$	40.8	**38.2**	25.8	29.2	26.8	32.6	28.1	24.5	25.1	25.7	25.2	2.5	20.7
Struct to Text$_{Rouge-l}$	58.7	**56.0**	42.9	45.9	43.2	50.8	45.4	41.1	41.9	42.7	42.2	6.4	34.6
Translation$_{BLEU}$	12.9	12.9	12.7	**14.6**	14.1	**14.6**	14.1	11.8	12.4	11.9	12.4	0.8	10.2
COMMONSENSE	69.5	59.0	47.5	61.0	56.0	64.0	60.5	65.0	**66.0**	64.0	61.0	17.5	34.0
SENTIMENT	90.0	90.5	91.0	87.0	83.5	**91.5**	91.5	90.0	89.5	90.0	89.0	79.5	11.0
READING Comp.	76.0	**60.3**	48.0	56.7	49.3	**60.3**	51.3	53.7	53.3	52.3	51.3	48.7	3.3
CLOSE-BOOK QA	64.0	60.0	53.0	62.0	58.0	**63.0**	61.0	59.5	57.5	58.5	57.5	34.5	6.5
COREFERENCE	74.0	75.0	65.0	55.0	59.0	**76.0**	64.0	61.0	62.0	56.0	57.0	55.0	10.0
READ. COOMP. W/ COM	82.0	**80.0**	33.0	57.0	49.0	78.0	58.0	51.0	48.0	49.0	49.0	13.0	14.0
PARAPHRASE	77.5	68.0	52.5	55.5	45.5	**71.0**	55.5	50.0	52.5	47.5	52.0	64.0	2.5
NLI	82.4	**78.9**	70.2	69.8	66.4	78.1	75.7	67.7	71.0	67.4	66.6	67.5	14.9

Table 1: We report the average performance of each task cluster. The full results of each task are shown in Appendix.C. "IID" signifies that LoraRetriever can access any LoRA for every test sample, encompassing the LoRA specific to the sample's task. "OOD" indicates that for each test sample, we mask the LoRA associated with its specific task during the retrieval phase. Consequently, no sample can access its ideal LoRA, allowing us to assess the LoraRetriever's cross-task generalization capability. The performance of perfectly selected corresponding LoRA for each sample is colored in gray. We have bolded the best performance of each task and underlined the best performance in the "OOD" setting.

Method	**Top 1**	**Top 3**	**Top 5**	**Top 8**
all-mpnet-base-v2	58.40	78.26	84.77	90.24
all-MiniLM-L6-v2	51.73	73.11	80.54	87.18
msmarco-distilbert-cos-v5	45.84	66.01	75.14	82.67
gtr-t5-xl	53.19	69.72	77.41	83.59
LoraRetriever $^{0\%}$	60.80	79.29	85.57	91.58
LoraRetriever $^{40\%}$	63.16	89.09	95.45	98.97
LoraRetriever $^{100\%}$	**74.08**	**97.37**	**99.15**	**99.82**

Table 2: Comparison of Sentence Embedding Techniques in LoRA Retrieval: The notation LoraRetriever$^{k\%}$ signifies that the model underwent supplementary training on k percent of the tasks. The performance of the selected retriever model in the evaluation phase is highlighted in gray.

fine-tuning based on Instructor-xl (Su et al., 2022). The training data consisted of only 40% of the tasks used to train task-specific LoRAs, with each task represented by 20 samples randomly selected from its respective LoRA training set. In this process, we categorized samples from the same LoRA as positive examples, while those from different LoRAs were considered negative examples. Additionally, the three distinct strategies for LoRA composition are: (1) **Selection**, which involves choosing the highest-ranked (top-1) retrieved LoRA and applying it in a singular LoRA manner, and can be viewed as a variant for Mixture and Fusion methods; (2) **Mixture**, which averaging the outputs of each submodule from the top-k retrieved LoRAs; and (3) **Fusion**, a method that averages the parameters of the top-k retrieved LoRAs. Throughout our experiments, $k = 3$ is established as the default setting.

Methods	0%	40% (Δ%)	100% (Δ%)
Selection	57.99	62.42 (+7.64%)	64.01 (+2.55%)
Fusion	51.50	51.50 (+0.00%)	52.27 (+1.49%)
Mixture	62.24	63.54 (+2.09%)	64.19 (+1.02%)

Table 3: Average Performance of LoraRetriever on NLU Tasks Across Different LoRA Composition Strategies and Task Training Percentages.

Metrics. Following Wei et al. (2021), we assess the performance on the "Struct to Text" task using Rouge-{1, 2, L} and on the "Translation" tasks using BLEU. Additionally, for the NLU tasks, we evaluate the exact match accuracy of each method.

5.2 Main Results

The main results of the mixed-task evaluation are shown in Tab.1. We present the mean performance across each task cluster and additionally evaluate the LoraRetriever's effectiveness in an out-of-domain (OOD) setting. In the OOD configuration, we mask the corresponding LoRA for each sample, thereby inhibiting LoraRetriever from retrieving the ideal LoRA for each sample. In this way, we can assess the cross-task generalization capability of LoraRetriever. From the results, we have the following observations: (1) The proposed framework, LoRARetriever, which performs input-aware LoRA retrieval and composition, markedly surpasses other baselines focusing on specific downstream tasks. (2) Among them, the performance of Mixture and Selection is similar in IID scenarios, while Fusion's performance is weaker compared to the other two methods. The reasons are as follows: (i) In the IID setting, LoraRetriever can achieve strong top-1 selection, leading to similar results between Selection and the Mixture; (ii) As different tasks are inherently heterogeneous, it is inferior to directly average top-k LoRA parameters in the Fusion. (3) In the OOD setting, the Mixture exceeds the performance of the Selection, and the performance of Fusion is similar to that of the Selection. The reasons can be as follows: (i) The selection cannot retrieve the associated LoRA for the input sample in the OOD setting, leading to a significant performance drop. (ii) The Mixture can fully leverage the capabilities of similar tasks to address OOD tasks, alleviating the performance drop. (4) The performance of the MoE and SMEAR methods is weaker than that of LoraRetriever. The limitation stems from the restricted capacity of these methods for adaptation and generalization to dynamically changing environments populated with diverse LoRAs, thereby diminishing their efficacy in mixed-task scenarios. (5) In mixed-task scenarios, although AdapterSoup uniformly searches for appropriate LoRAs for downstream tasks, the retrieved LoRAs fall short in personalization for each request, hindering their effectiveness for each specific task. (6) LoRAhub proves to be entirely ineffective in the mix-task scenario. First, the fusion of LoRAhub depends on randomly selected LoRAs, which may not be relevant. Second, the presence of heterogeneous tasks introduces conflicting parameter optimization directions, resulting in the total breakdown of parameter fusion.

5.3 Analysis

Performance of Retriever. We compare LoraRetriever with some popular off-the-shelf sentence embedding models in Huggingface and adopt the model nomenclature following Wolf et al. (2020). To analyze the effect of the percentage of the tasks for training LoraRetriever, we trained three variants of LoraRetriever with different percentages. Tab.2 shows the performance of different retrieval models for retrieving relevant LoRAs. It is shown that guiding sentence embedding models with specific prompts leads to a performance improvement in retrieval compared to common retrieval models. After instruction fine-tuning, the retriever significantly enhanced the ability to retrieve corresponding LoRA based on the input. Conducting instruction fine-tuning on 40% of the tasks resulted in a 2.36% increase in top-1 accuracy and a 9.80% increase in top-3 accuracy. Training across all tasks achieved the largest improvement. To demonstrate the generalizability of the proposed framework when dealing with unseen LoRAs, we used a retriever trained on 40% of the tasks in the main experiment to simulate the scenario of dynamic updates to the LoRA pool that might occur while providing services with LoraRetriever.

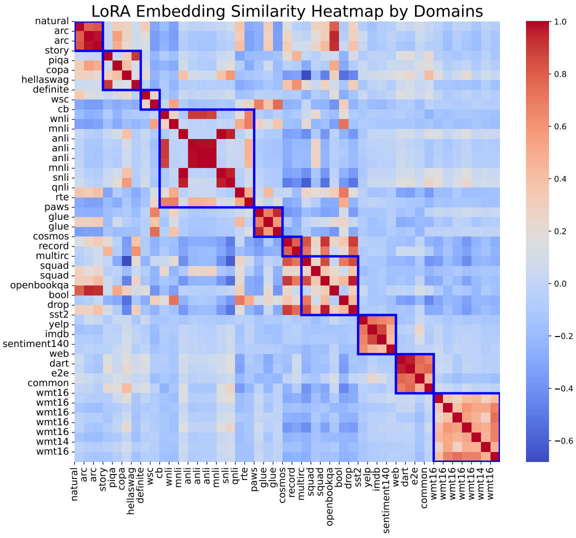
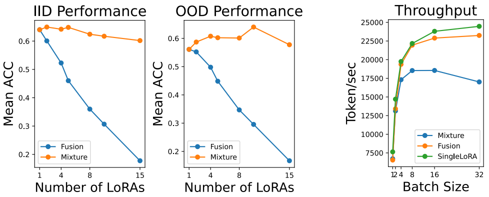


Figure 3: LoRA embedding similarity heatmap. Tasks from the same domain are grouped in square brackets.

Tab.3 shows the performance of LoraRetriever trained with different proportions of tasks for LoRA retrieval. It is observed that for the selection and mixture methods, training under 40% of the tasks has already seen significant improvements. The best performance is achieved when trained under all tasks, but the improvement compared to 40% is relatively small, which to some extent reflects the good generalization ability of instruction fine-tuning of LoraRetriever.

Fig.3 illustrates the similarity between task em-



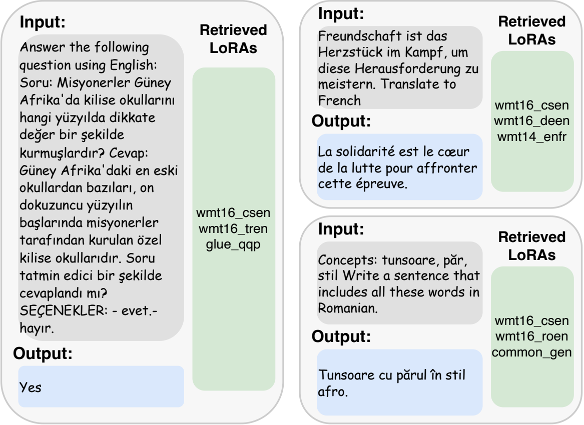
a) Performance with Different Number of LoRAs b) Throughput Varyinig Batch Size

Figure 4: The left figure shows the performance of LoraRetriever varying the number of LoRAs. The right figure shows the performance of Throughput varying the batch size.

beddings for different tasks through a heatmap, where tasks from the same task cluster are grouped in square brackets. It is shown that task embeddings within the same domain are more similar, indicating that the LoraRetriever embeddings can serve as task embeddings to characterize the similarities between different tasks and be applied for LoRA retrieval.

Impact of the number of Retrieved LoRA. Fig.4 (a) illustrates the performance of the number of retrieved LoRAs on the mean accuracy of the NLU tasks. The results indicate that as the number of retrieved LoRAs increases, the performance of the Mixture initially improves slightly but then stabilizes. In contrast, the Fusion shows a continuous decline in performance with an increasing number of LoRAs, which once again demonstrates that under the conditions of heterogeneous tasks, the simple averaging of parameters can compromise the original capabilities of the LoRAs. In particular, in the OOD setting, the performance of the Mixture improves significantly as the number of LoRAs increases, illustrating that in the absence of an ideal LoRA choice for a request, leveraging the capabilities of multiple LoRAs of similar tasks can effectively achieve cross-task generalization.

Effectiveness of Batch Inference Strategy. To evaluate the efficiency of our proposed batch inference strategy, we compared the throughput of different batch sizes. The throughput is defined as the number of both input and output tokens per second across all requests in the mixed-task benchmark. We specifically compared the computational efficiency with that of a single LoRA. Our evaluation encompassed the entire evaluation dataset, and we limited the generation to the first produced token to mitigate discrepancies caused by varying generation lengths across different methods. These experiments were conducted on an NVIDIA A100 GPU



Input:

Answer the following question using English: Soru: Misyonerler Güney Afrika'da kilise okullarını hangi yüzyılda dikkate değer bir şekilde kurmuşlardır? Cevap: Güney Afrika'daki en eski okullardan bazıları, on dokuzuncu yüzyılın başlarında misyonerler tarafından kurulan özel kilise okullarıdır. Soru tatmin edici bir şekilde cevaplandı mı? SEÇENEKLER: - evet.- hayır.

Output:

Yes

Retrieved LoRAs

wmt16_csen
wmt16_tren
glue_qqp

Input:

Freundschaft ist das Herzstück im Kampf, um diese Herausforderung zu meistern. Translate to French

Output:

La solidarité est le cœur de la lutte pour affronter cette épreuve.

Retrieved LoRAs

wmt16_csen
wmt16_deen
wmt14_enfr

Input:

Concepts: tunsoare, păr, stil Write a sentence that includes all these words in Romanian.

Output:

Tunsoare cu părul în stil afro.

Retrieved LoRAs

wmt16_csen
wmt16_roen
common_gen

Figure 5: Showcasing How the LoRARetrieval Framework Employs Multiple LoRAs for Cooperative Problem Solving.

(80GB) utilizing bfloat16 precision. As illustrated in Fig.4 (b), our batch inference strategy markedly improves the throughput of the framework, with a slight throughput reduction compared to a single LoRA. Notably, the Fusion outperforms the mixture strategy in throughput efficiency, attributed to its parameter averaging approach that circumvents the need for parallel computation across multiple LoRAs.

Showcases. We showcase the framework's ability to adeptly integrate multiple LoRAs for synergistic problem-solving, as evidenced in Fig.5. We manually craft three problems in Fig.5, which cannot retrieve any single LoRA to solve these problems directly, necessitating the cooperation of existing LoRAs. Specifically, the first example requires LoraRetriever to integrate NLI and translation tasks' capabilities. The retrieved LoRA wmt16-tren is utilized for comprehending Turkish, while glue-qqp is applied to NLI tasks. In the second scenario, LoRAs are integrated for translating from German to French. Although there is no direct LoRA for German-to-French translation, the combined use of wmt16-deen for German-to-English and wmt14-enfr for English-to-French enables an effective German-to-French translation. The third scenario illustrates the fusion of distinct capabilities by combining Romanian translation with text generation: leveraging the wmt16-roen LoRA for Romanian comprehension and the common-gen LoRA for generating text, LoraRetriever successfully merges these diverse functionalities. This demonstration emphasizes the framework's substantial ability to blend distinct LoRA capabilities, anticipating further exploration of capability fusion of LoRAs as a future direction.

6 Conclusion

This paper investigates a new problem of serving multiple LoRAs with a dynamically updated LoRA pool for downstream heterogeneous requests. To this end, we introduce a framework named LoraRetriver to identify and retrieve the appropriate LoRAs based on a specific input. Subsequently, we focus on the composition of these retrieved LoRAs to ensure a tailored and practical application in real-world situations. We also propose an efficient batch inference strategy to accommodate batched requests. Subsequent experiments have also demonstrated the effectiveness of our proposed LoraRetriever.

Limitation

While promising, there are still some drawbacks of LoraRetriever. (1) User data privacy issues. When users upload LoRA, we need to use a small amount of training data (10-20 pieces) to represent the distribution of the LoRA model. In privacy-sensitive scenarios, representation with data may not be feasible. Aligning LoRA parameters and sample distributions in the embedding space in a manner that respects data privacy presents a worthwhile direction for future exploration. (2) The proposed LoraRetriever framework is only suitable for multi-LoRA collaboration under the same model architecture. However, in reality, the model architecture chosen by the users themselves and the PEFT method are not necessarily the same, which is worth further research on how to design the corresponding collaborative mechanism for such scenarios.

Ethics Statement

In this work, we take the open-sourced Llama-2 (Touvron et al., 2023) model and Flan-v2 (Wei et al., 2021) dataset in the experiment. One potential ethical concern may arise when applying the proposed framework to real-world scenarios, as the data uploaded by LoRA contributors may contain user privacy. However, we believe this can be addressed by proper anonymization before uploading. Therefore, we believe our work will not pose a severe ethical concern.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (62441605, 62376243, 62037001, U20A20387), and the Starry Night Science Fund of Zhejiang University Shanghai Institute for Advanced Study (SN-ZJU-SIAS-0010).

References

Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. 2019. Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6430–6439.

Anonymous. 2024. MoLE: Mixture of loRA experts. In *The Twelfth International Conference on Learning Representations*.

Akari Asai, Timo Schick, Patrick Lewis, Xilun Chen, Gautier Izacard, Sebastian Riedel, Hannaneh Hajishirzi, and Wen-tau Yih. 2022. Task-aware retrieval with instructions. *arXiv preprint arXiv:2211.09260*.

Ziwei Chai, Guoyin Wang, Jing Su, Tianjie Zhang, Xuanwen Huang, Xuwu Wang, Jingjing Xu, Jianbo Yuan, Hongxia Yang, Fei Wu, et al. 2024. An expert is worth one token: Synergizing multiple expert llms as generalist via expert token routing. *arXiv preprint arXiv:2403.16854*.

Alexandra Chronopoulou, Matthew E Peters, Alexander Fraser, and Jesse Dodge. 2023. Adaptersoup: Weight averaging to improve generalization of pretrained language models. *arXiv preprint arXiv:2302.07027*.

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. 2023. Loramoe: Revolutionizing mixture of experts for maintaining world knowledge in language model alignment.

Muhammad Usman Hadi, Rizwan Qureshi, Abbas Shah, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, Seyedali Mirjalili, et al. 2023. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*.

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*.

Zhiqiang Hu, Yihuai Lan, Lei Wang, Wanyu Xu, Ee-Peng Lim, Roy Ka-Wei Lee, Lidong Bing, and Soujanya Poria. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*.

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

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Michael I Jordan and Robert A Jacobs. 1994. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

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

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2023b. Gpt understands, too. *AI Open*.

Mohammed Muqeeth, Haokun Liu, and Colin Raffel. 2023. Soft merging of experts with adaptive routing. *arXiv preprint arXiv:2306.03745*.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. 2021. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877*.

Xiaonan Nie, Shijie Cao, Xupeng Miao, Lingxiao Ma, Jilong Xue, Youshan Miao, Zichao Yang, Zhi Yang, and CUI Bin. 2021. Dense-to-sparse gate for mixture-of-experts.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. 2021. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595.

Sheng Shen, Le Hou, Yanqi Zhou, Nan Du, Shayne Longpre, Jason Wei, Hyung Won Chung, Barret Zoph, William Fedus, Xinyun Chen, et al. 2023. Mixture-of-experts meets instruction tuning: A winning combination for large language models. *arXiv preprint arXiv:2305.14705*.

Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, et al. 2023. S-lora: Serving thousands of concurrent lora adapters. *arXiv preprint arXiv:2311.03285*.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A Smith, Luke Zettlemoyer, and Tao Yu. 2022. One embedder, any task: Instruction-finetuned text embeddings. *arXiv preprint arXiv:2212.09741*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2023. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*.

Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. Adamix: Mixture-of-adaptations for parameter-efficient model tuning. *arXiv preprint arXiv:2210.17451*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Yeming Wen and Swarat Chaudhuri. 2023. Batched low-rank adaptation of foundation models. *arXiv preprint arXiv:2312.05677*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Yiquan Wu, Siying Zhou, Yifei Liu, Weiming Lu, Xiaozhong Liu, Yating Zhang, Changlong Sun, Fei Wu, and Kun Kuang. 2023. Precedent-enhanced legal judgment prediction with llm and domain-model collaboration. *arXiv preprint arXiv:2310.09241*.

Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. 2023. Fingpt: Open-source financial large language models. *arXiv preprint arXiv:2306.06031*.

Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermiş, Acyr Locatelli, and Sara Hooker. 2023. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. *arXiv preprint arXiv:2309.05444*.

Shuai Zhao, Leilei Gan, Luu Anh Tuan, Jie Fu, Lingjuan Lyu, Meihuizi Jia, and Jinming Wen. 2024. Defending against weight-poisoning backdoor attacks for parameter-efficient fine-tuning. *arXiv preprint arXiv:2402.12168*.

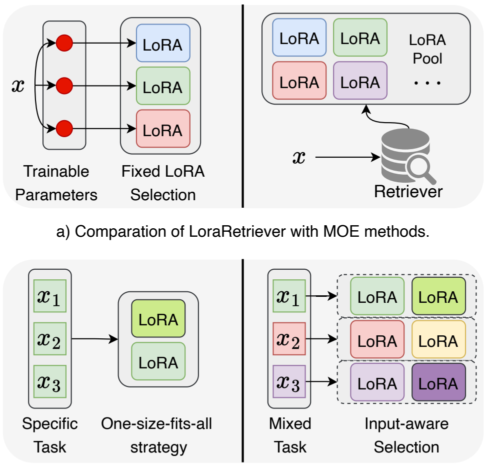
Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: Fast and robust inference with early exit. In *Advances in Neural Information Processing Systems*, volume 33, pages 18330–18341. Curran Associates, Inc.

Wangchunshu Zhou, Canwen Xu, and Julian McAuley. 2022. Efficiently tuned parameters are task embeddings. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5007–5014, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yun Zhu, Nevan Wichers, Chu-Cheng Lin, Xinyi Wang, Tianlong Chen, Lei Shu, Han Lu, Canoee Liu, Liangchen Luo, Jindong Chen, et al. 2023. Sira: Sparse mixture of low rank adaptation. *arXiv preprint arXiv:2311.09179*.

A Details of Baseline Methods

(1) **Mixture of Experts** (Zhu et al., 2023; Zadouri et al., 2023; Liu et al., 2023a; Wang et al., 2022; Anonymous, 2024). Many works have considered coordinating different adapters through MoE, and here we explored three distinct variants: one employing a soft mixture of experts and the other utilizing discrete routing (top1 and top3). Detailed information on the training aspects of MoE methods is provided in the Appendix.F. (2) **SMEAR** (Muqeeth et al., 2023) introduces the concept of adaptive routing by performing a weighted average of different adapters' parameters to utilize various experts effectively. For the MoE and SMEAR baselines, challenges arise in scaling due to training confined to a limited set of LoRAs. Consequently, we strategically selected a dedicated LoRA expert for each domain to specialize in router training. (3) **AdapterSoup** (Chronopoulou et al., 2023) uniformly selects the corresponding LoRAs for the entire downstream task, which lacks the ability to provide personalized service for diverse requests. (4) **LoRAhub** (Huang et al., 2023) enables black-box optimization to learn the weights of various LoRA parameters, thereby facilitating weighted parameter averaging for specific downstream tasks. In our implementation, we conformed to the default setting, which entails randomly selecting 20 LoRAs from the available LoRA pool and performing weighted parameter averaging. For the MoE, SMEAR, and LoRAhub approaches, we selected 20 data samples from the training datasets of all tasks to serve as their training data.



a) Comparation of LoraRetriever with MOE methods.

b) Comparation of LoraRetriever with LoRAhub&AdapterSoup.

Figure 6: The advantages of LoraRetriever compared to previous methods. The above figure shows the difference between our method and MoE in expert routing; the figure below demonstrates the difference between our method's focus on mixed-task scenarios and previous methods that only targeted specific tasks.

B Main Difference between LoraRetriever and Baseline Methods

As depicted in Fig.6, LoraRetriever distinguishes itself from baseline methods significantly. It utilizes an off-the-shelf retriever for routing experts, contrasting with MoE's approach, to achieve not only more flexibility but also enhanced generalization to unseen LoRAs. Moreover, unlike isolated task-focused methods like LoRAhub and AdapterSoup, LoraRetriever is adept at adaptively managing heterogeneous downstream tasks.

C Full Results

In Tab.4, we show the full results of the mixed-task scenario of all tasks.

D PyTorch-style pseudocode for batch inference

The batch inference process can be easily achieved through a few lines of einsum operation. We show the PyTorch style pseudocode in Alg.1.

Algorithm 1: PyTorch-style pseudocode for batch inference

```
# X: (b,l,d), M: (b,p)
# A: (p,r,d), B: (p,d,r)
# LoRA fusion computation
FA=torch.einsum(’bp,prd->brd’,M,A)
FB=torch.einsum(’bp,pdr->bdr’,M,B)
mid=torch.einsum(’bld,brd->blr’,X,FA)
res=torch.einsum(’blr,bdr->bld’,mid,FB)
# LoRA mixture computation
mid=torch.einsum(’bld,prd->blpr’,X,A)
mid=torch.einsum(’blpr,pdr->blpd’,mid,B)
res=torch.einsum(’blpd,bp->bld’,mid,M)
```

E Details of Training and Evaluation datasets

We leverage a subset of flan-v2 datasets (Wei et al., 2021) as shown in Fig.7 for LoRA expert training and mixed-task dataset generation. We summarize the details of the used datasets as follows:

Struct-to-Text Conversion: This task evaluates the capability to generate natural language descriptions from structured data inputs. We use the following datasets: (1) CommonGen; (2) DART; (3) E2ENLG; (4) WebNLG;

Translation: Translation involves converting text from one language to another, maintaining the original meaning and nuances. We use the following datasets: (1) En-Fr from WMT'14; En-De, En-Tr, En-Ru, En-Fi, En-Ro from WMT'16; (3) En-Es from Paracrawl.

Commonsense Reasoning: This involves assessing the ability to apply physical or scientific principles alongside common sense in reasoning tasks. We use the following datasets: (1) COPA, (2) HellaSwag, (3) PiQA, and (4) StoryCloze.

Sentiment Analysis: A fundamental task in natural language processing (NLP) that determines the sentiment polarity (positive or negative) of a given text. We use the following datasets: (1) IMDB, (2) Sentiment140, (3) SST-2, and (4) Yelp.

Closed-Book Question Answering: This task challenges models to answer questions about general knowledge without direct access to external information sources. We use the following datasets: (1) ARC, (2) NQ, and (3) TriviaQA.

Paraphrase Detection: This task requires models to ascertain whether two sentences convey the same meaning, indicating semantic equivalence. We use the following datasets: (1) MRPC, (2) QQP, and (3) Paws Wiki.

Coreference Resolution: Involves identifying instances within a text that refer to the same entity, demonstrating an understanding of textual context. We use the following datasets: (1) DPR and (2) WSC273.

Reading comprehension: Assesses the capability to derive answers to questions from a provided text containing relevant information. We use the following datasets: (1) BoolQ, (2) DROP, (3) MultiRC, (4) OBQA, (5) SQuADv1, (6) SQuADv2.

Reading Comprehension with Commonsense: Merges traditional reading comprehension skills with commonsense reasoning, requiring understanding beyond the explicit text. We use the following datasets: (1) CosmosQA; (2) ReCoRD.

Natural Language Inference: Focuses on deducing the relationship between two sentences, determining if the second sentence logically follows from, contradicts, or is unrelated to the first sentence. We use the following datasets: (1) ANLI, (2) CB; (3) MNLI; (4) QNLI; (5) SNLI; (6) WNLI; (7) RTE.

F Implementation Details of Baseline Methods

F.1 MoE baselines

We use E to denote the LoRA expert and R to denote the router. The MoE methods can be expressed

Task / Llama-2-7b	Perfect	Selection		Fusion		Mixture		MoE	MoE	MoE	SME-	Adapter	LoRA
	Selection	IID	OOD	IID	OOD	IID	OOD	Top1	Top3	Soft	AR	Soup	Hub
Struct to Text													
WebNLG $^{\text{Rouge-1}}$	71.2	67.0	53.9	49.4	45.4	57.8	53.9	45.1	47.6	49.1	51.1	3.9	32.5
WebNLG $^{\text{Rouge-2}}$	50.6	44.5	30.0	25.9	24.1	33.5	29.4	22.6	25.8	26.1	27.9	0.9	17.3
WebNLG $^{\text{Rouge-l}}$	64.4	60.9	49.1	45.5	41.0	52.3	49.6	40.0	41.9	43.3	45.4	3.9	31.1
DART $^{\text{Rouge-1}}$	71.7	67.9	58.4	56.3	53.4	63.2	60.0	55.4	56.3	56.9	60.0	3.3	40.0
DART $^{\text{Rouge-2}}$	49.1	45.8	34.9	32.3	30.6	36.6	35.4	30.3	31.0	30.8	33.0	1.3	20.1
DART $^{\text{Rouge-l}}$	64.6	61.1	52.4	50.3	47.9	56.3	52.4	49.7	50.8	50.2	54.8	3.3	35.2
E2ENLG $^{\text{Rouge-1}}$	66.1	65.8	59.3	62.2	57.2	66.0	58.7	52.9	54.0	55.3	53.2	4.2	50.1
E2ENLG $^{\text{Rouge-2}}$	40.0	39.4	34.1	34.7	32.0	38.8	32.1	26.9	27.6	28.8	27.5	2.4	26.3
E2ENLG $^{\text{Rouge-l}}$	56.7	55.7	50.2	52.7	49.1	56.9	49.0	45.1	45.0	47.0	45.1	4.2	42.2
CommonGen $^{\text{Rouge-1}}$	46.9	44.7	29.0	29.9	27.7	36.5	29.0	29.0	29.3	30.1	27.6	6.6	19.8
CommonGen $^{\text{Rouge-2}}$	18.8	18.3	7.3	9.9	7.2	11.1	8.6	7.7	7.1	9.3	8.4	0.0	6.9
CommonGen $^{\text{Rouge-l}}$	42.5	40.5	24.0	25.8	23.3	32.7	24.8	24.4	25.1	26.3	24.3	6.6	18.0
Translation													
Paracrawl-enes	24.3	24.2	20.3	22.9	22.3	22.8	22.1	18.0	18.8	19.5	21.6	4.5	16.4
WMT'16-tren	3.2	3.1	2.6	3.5	3.3	3.7	2.6	3.5	3.2	3.4	3.2	0.0	2.0
WMT'16-ruen	10.8	10.4	9.8	9.2	9.3	11.0	10.8	6.2	7.8	8.3	7.3	0.0	4.8
WMT'16-deen	18.9	18.7	20.3	17.9	18.8	18.8	18.7	11.6	14.0	14.7	16.6	1.1	11.4
WMT'16-fien	6.5	6.5	7.0	7.2	7.1	7.3	7.8	6.2	6.2	6.1	6.5	0.7	4.3
WMT'16-roen	13.9	14.0	12.3	12.8	13.3	13.1	12.2	9.8	10.7	10.1	10.3	0.3	8.0
WMT'14-enfr	16.5	16.1	16.9	17.7	18.0	17.8	18.0	15.9	17.3	17.1	16.4	3.5	15.2
WMT'16-csen	10.7	9.4	7.0	6.1	6.2	8.3	5.8	4.7	6.3	6.3	6.3	0.8	6.1
COMMONSENSE													
StoryCloze	72.0	62.0	42.0	72.0	68.0	84.0	58.0	74.0	70.0	70.0	68.0	62.0	48.0
PIQA	46.0	46.0	32.0	34.0	36.0	38.0	34.0	40.0	38.0	38.0	36.0	38.0	0.0
COPA	86.0	74.0	68.0	78.0	70.0	80.0	68.0	72.0	70.0	72.0	70.0	56.0	22.0
HellaSwag	46.0	40.0	42.0	20.0	18.0	44.0	40.0	32.0	30.0	26.0	26.0	28.0	0.0
sentiment													
SST-2	98.0	98.0	96.0	74.0	78.0	96.0	94.0	56.0	68.0	66.0	66.0	74.0	0.0
Yelp	98.0	94.0	94.0	96.0	96.0	98.0	98.0	86.0	90.0	86.0	84.0	80.0	0.0
IMDB	96.0	96.0	96.0	92.0	82.0	96.0	96.0	76.0	80.0	80.0	84.0	80.0	0.0
sentiment140	68.0	70.0	70.0	54.0	58.0	68.0	74.0	62.0	62.0	66.0	62.0	60.0	2.0
READING Comp.													
MultiRC	68.0	52.0	38.0	44.0	44.0	48.0	44.0	54.0	52.0	50.0	48.0	40.0	6.0
SQuADv2	62.0	56.0	12.0	30.0	20.0	22.0	16.0	24.0	24.0	26.0	22.0	16.0	0.0
SQuADv1	68.0	66.0	68.0	64.0	64.0	62.0	68.0	68.0	70.0	66.0	66.0	54.0	4.0
OBQA	82.0	68.0	58.0	64.0	60.0	78.0	66.0	62.0	64.0	66.0	60.0	40.0	0.0
BoolQ	84.0	60.0	60.0	68.0	70.0	80.0	76.0	74.0	68.0	76.0	70.0	72.0	6.0
drop	40.0	8.0	6.0	14.0	12.0	18.0	14.0	10.0	8.0	8.0	8.0	22.0	0.0
CLOSE-BOOK QA													
NQ	18.0	16.0	10.0	16.0	14.0	16.0	10.0	12.0	12.0	12.0	4.0	12.0	0.0
ARC-e	50.0	56.0	70.0	54.0	56.0	66.0	82.0	58.0	58.0	60.0	58.0	48.0	0.0
ARC-c	46.0	42.0	46.0	34.0	34.0	50.0	46.0	46.0	42.0	42.0	42.0	24.0	0.0
TriviaQa	66.0	46.0	46.0	60.0	46.0	48.0	56.0	46.0	42.0	46.0	24.0	42.0	4.0
COREFERENCE													
DPR	54.0	50.0	50.0	56.0	60.0	68.0	56.0	64.0	60.0	62.0	62.0	46.0	2.0
WSC	50.0	50.0	42.0	38.0	46.0	58.0	42.0	58.0	58.0	52.0	54.0	40.0	0.0
READ. COOMP. W/ COMMONSENSE													
CosmosQa	68.0	68.0	34.0	46.0	32.0	50.0	46.0	44.0	46.0	44.0	38.0	14.0	6.0
record	70.0	70.0	26.0	24.0	6.0	42.0	34.0	18.0	12.0	14.0	8.0	14.0	0.0
PARAPHRASE													
Paws Wiki	90.0	64.0	40.0	44.0	42.0	56.0	46.0	56.0	50.0	48.0	54.0	60.0	2.0
QQP	74.0	74.0	68.0	66.0	60.0	80.0	58.0	50.0	40.0	36.0	28.0	54.0	0.0
MRPC	60.0	58.0	58.0	60.0	62.0	60.0	58.0	42.0	44.0	40.0	42.0	60.0	2.0
STSB	38.0	36.0	16.0	12.0	12.0	30.0	20.0	20.0	20.0	20.0	14.0	12.0	0.0
NLI													
CB	88.9	80.0	62.2	77.8	57.8	86.7	66.7	68.9	64.4	68.9	62.2	55.6	13.3
WNLI	70.0	68.0	46.0	44.0	50.0	60.0	54.0	56.0	56.0	42.0	44.0	52.0	0.0
ANLI-r1	50.0	50.0	50.0	40.0	42.0	40.0	42.0	40.0	40.0	36.0	38.0	38.0	24.0
ANLI-r2	46.0	46.0	46.0	32.0	36.0	46.0	46.0	40.0	36.0	38.0	32.0	46.0	20.0
ANLI-r3	46.0	42.0	38.0	38.0	40.0	44.0	50.0	28.0	32.0	34.0	38.0	40.0	24.0
MNLI-m	88.0	84.0	88.0	62.0	66.0	80.0	88.0	48.0	54.0	50.0	56.0	76.0	0.0
MNLI-mm	92.0	90.0	94.0	64.0	82.0	88.0	90.0	48.0	48.0	50.0	60.0	84.0	2.0
SNLI	96.0	84.0	84.0	56.0	58.0	90.0	92.0	54.0	52.0	54.0	54.0	82.0	0.0
QNLI	94.0	94.0	26.0	46.0	48.0	74.0	38.0	56.0	56.0	54.0	60.0	70.0	0.0
RTE	52.0	62.0	72.0	54.0	58.0	70.0	76.0	64.0	58.0	56.0	64.0	80.0	22.0

Table 4: Mixed Tasks evaluation on both NLU & NLG tasks. "OOD" indicates that during retrieval, we masked the corresponding task's LoRA for testing generalization when facing unknown tasks.

Task / Llama-2-13b	Perfect	Selection		Fusion		Mixture		MoE	MoE	MoE	SME-	Adapter	LoRA
	Selection	IID	OOD	IID	OOD	IID	OOD	Top1	Top3	Soft	AR	Soup	Hub
Struct to Text													
WebNLG $^{\text{Rouge-1}}$	72.6	68.5	51.9	55.2	51.3	59.7	53.7	47.0	47.8	48.3	49.7	6.6	34.2
WebNLG $^{\text{Rouge-2}}$	51.4	47.5	28.6	30.1	27.4	35.6	29.1	25.5	26.4	27.0	26.3	3.2	16.8
WebNLG $^{\text{Rouge-l}}$	66.0	62.4	48.3	49.4	46.2	55.1	49.0	42.5	44.3	43.5	44.3	6.3	32.4
DART $^{\text{Rouge-1}}$	74.0	67.0	57.0	60.4	58.7	62.6	60.6	57.9	57.0	58.7	58.9	12.5	43.3
DART $^{\text{Rouge-2}}$	54.6	45.9	33.6	37.4	35.2	38.9	37.3	32.6	33.2	34.1	34.3	5.9	25.5
DART $^{\text{Rouge-l}}$	67.7	61.2	50.0	54.0	52.2	55.0	53.4	50.9	50.9	51.8	51.5	11.8	38.1
E2ENLG $^{\text{Rouge-1}}$	66.4	66.1	59.2	65.6	61.7	66.7	63.9	52.8	53.9	55.5	55.1	2.5	58.6
E2ENLG $^{\text{Rouge-2}}$	39.6	39.3	32.8	36.6	33.5	39.0	36.4	27.8	28.4	28.3	28.8	0.9	31.6
E2ENLG $^{\text{Rouge-l}}$	56.7	56.4	48.9	53.4	49.7	56.2	53.7	43.0	44.5	45.1	45.4	2.2	48.2
CommonGen $^{\text{Rouge-1}}$	48.5	48.9	29.3	29.5	27.0	41.7	30.0	29.5	29.3	31.5	29.5	6.9	21.2
CommonGen $^{\text{Rouge-2}}$	17.7	20.3	8.2	12.6	11.3	17.0	9.5	12.2	12.3	13.5	11.5	0.0	8.8
CommonGen $^{\text{Rouge-l}}$	44.3	44.1	24.4	26.7	24.6	36.7	25.3	28.0	27.9	30.3	27.8	5.4	19.7
Translation													
Paracrawl-enes	24.4	25.4	23.1	28.3	27.2	27.4	25.8	21.8	21.5	20.0	24.2	4.0	19.0
WMT'16-tren	2.9	2.4	1.2	3.7	3.2	3.4	2.7	3.0	3.3	2.7	3.3	0.0	1.9
WMT'16-ruen	11.8	11.5	10.3	11.8	11.5	10.5	12.0	8.8	9.9	9.3	8.8	0.0	8.1
WMT'16-deen	19.9	19.9	20.7	20.2	20.7	20.2	20.2	16.5	18.6	18.1	18.3	2.2	15.1
WMT'16-fien	7.3	6.8	5.1	9.8	7.3	8.7	7.8	8.0	8.3	8.4	8.3	0.0	6.3
WMT'16-roen	14.0	13.9	10.9	11.6	11.1	17.4	13.3	7.9	9.1	9.3	9.3	0.0	7.4
WMT'14-enfr	16.6	17.1	18.4	20.7	21.1	18.7	20.9	17.4	17.5	18.7	17.2	0.4	15.1
WMT'16-csen	6.6	6.2	11.6	11.1	10.5	10.3	10.1	10.6	10.7	9.0	9.4	0.0	8.8
COMMONSENSE													
StoryCloze	96.0	80.0	56.0	90.0	76.0	80.0	76.0	96.0	98.0	94.0	92.0	18.0	64.0
PIQA	48.0	52.0	30.0	46.0	46.0	46.0	46.0	42.0	40.0	36.0	38.0	14.0	10.0
COPA	76.0	74.0	68.0	74.0	74.0	78.0	76.0	72.0	80.0	80.0	76.0	22.0	60.0
HellaSwag	58.0	30.0	36.0	34.0	28.0	52.0	44.0	50.0	46.0	46.0	38.0	16.0	2.0
sentiment													
SST-2	98.0	98.0	98.0	86.0	86.0	98.0	100.0	98.0	96.0	98.0	96.0	82.0	26.0
Yelp	98.0	98.0	100.0	94.0	92.0	98.0	98.0	98.0	98.0	98.0	98.0	82.0	0.0
IMDB	96.0	98.0	98.0	88.0	82.0	98.0	98.0	100.0	100.0	100.0	98.0	90.0	4.0
sentiment140	68.0	68.0	68.0	80.0	74.0	72.0	70.0	64.0	64.0	64.0	64.0	64.0	14.0
READING Comp.													
MultiRC	88.0	72.0	36.0	64.0	42.0	66.0	44.0	44.0	48.0	40.0	40.0	72.0	2.0
SQuADv1	74.0	62.0	62.0	58.0	58.0	60.0	60.0	64.0	66.0	64.0	60.0	42.0	8.0
SQuADv2	66.0	58.0	34.0	38.0	24.0	34.0	24.0	26.0	22.0	24.0	24.0	36.0	0.0
OBQA	86.0	80.0	72.0	76.0	76.0	88.0	82.0	80.0	76.0	78.0	76.0	48.0	0.0
BoolQ	84.0	68.0	66.0	84.0	82.0	78.0	78.0	86.0	84.0	88.0	86.0	74.0	10.0
Drpo	58.0	22.0	18.0	20.0	14.0	36.0	20.0	22.0	24.0	20.0	22.0	20.0	0.0
CLOSE-BOOK QA													
NQ	30.0	28.0	12.0	22.0	14.0	24.0	16.0	12.0	10.0	12.0	10.0	6.0	2.0
ARC-e	90.0	90.0	88.0	92.0	92.0	94.0	94.0	90.0	90.0	86.0	90.0	58.0	8.0
ARC-c	68.0	66.0	58.0	64.0	60.0	68.0	70.0	68.0	60.0	68.0	62.0	38.0	4.0
TriviaQa	68.0	56.0	54.0	70.0	66.0	66.0	64.0	68.0	70.0	68.0	68.0	36.0	12.0
COREFERENCE													
DPR	90.0	90.0	72.0	68.0	66.0	88.0	72.0	76.0	76.0	68.0	72.0	52.0	20.0
WSC	58.0	60.0	58.0	42.0	52.0	64.0	56.0	46.0	48.0	44.0	42.0	58.0	0.0
READ. COOMP. W/ COMMONSENSE													
CosmosQa	84.0	82.0	38.0	80.0	76.0	82.0	70.0	74.0	74.0	74.0	80.0	8.0	28.0
record	80.0	78.0	28.0	34.0	22.0	74.0	46.0	28.0	22.0	24.0	18.0	18.0	0.0
PARAPHRASE													
Paws Wiki	92.0	70.0	52.0	74.0	50.0	88.0	74.0	54.0	66.0	54.0	62.0	90.0	8.0
QQP	90.0	88.0	76.0	66.0	56.0	84.0	62.0	70.0	66.0	60.0	66.0	78.0	2.0
MRPC	84.0	70.0	62.0	64.0	64.0	78.0	64.0	62.0	62.0	62.0	64.0	82.0	0.0
STSB	44.0	44.0	20.0	18.0	12.0	34.0	22.0	14.0	16.0	14.0	16.0	6.0	0.0
NLI													
CB	100.0	91.1	84.4	84.4	80.0	93.3	88.9	73.3	82.2	75.6	77.8	73.3	13.3
WNLI	72.0	72.0	62.0	70.0	68.0	76.0	66.0	58.0	66.0	56.0	56.0	76.0	6.0
ANLI-r1	70.0	70.0	70.0	56.0	50.0	64.0	68.0	48.0	44.0	44.0	46.0	50.0	12.0
ANLI-r2	64.0	56.0	56.0	38.0	36.0	60.0	60.0	48.0	48.0	48.0	42.0	48.0	8.0
ANLI-r3	68.0	56.0	56.0	46.0	46.0	62.0	60.0	46.0	48.0	56.0	58.0	48.0	20.0
MNLI-m	88.0	90.0	88.0	88.0	88.0	86.0	88.0	90.0	94.0	88.0	80.0	96.0	4.0
MNLI-mm	90.0	90.0	94.0	94.0	92.0	94.0	100.0	94.0	98.0	88.0	88.0	80.0	2.0
SNLI	90.0	88.0	88.0	76.0	74.0	90.0	92.0	96.0	96.0	94.0	86.0	80.0	16.0
QNLI	94.0	94.0	30.0	68.0	56.0	74.0	58.0	60.0	62.0	60.0	56.0	56.0	32.0
RTE	88.0	82.0	74.0	78.0	74.0	82.0	76.0	64.0	72.0	64.0	76.0	68.0	36.0

Table 5: Mixed Tasks evaluation on both NLU & NLG tasks. “OOD" indicates that during retrieval, we masked the corresponding task’s LoRA for testing generalization when facing unknown tasks.

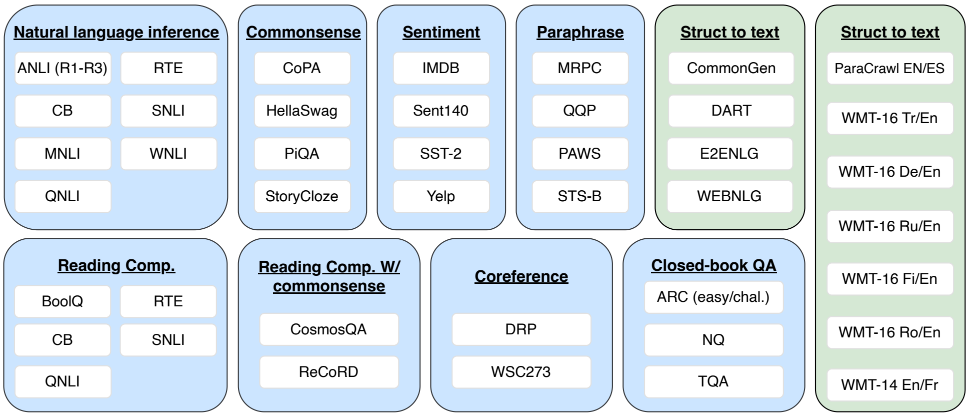


Figure 7: Datasets and task clusters used to train LoRAs and generate mixed-task evaluation set in this paper (NLU tasks in blue; NLG tasks in green).

in the following way:

$$y = \sum_{i=i}^{k} R(x)_i E_i(x). \tag{5}$$

We implied two variants of the MoE routing mechanism. (1) **Dense Gating.** Following (Zadouri et al., 2023), the router network consists of a dense layer with trainable parameter W_g, and the gating score could be obtained through a softmax function by:

$$s_i = R(x)_i = softmax(W_g^T x), \tag{6}$$

(2)**Sparse Gate**. To maintain the sparsity while training, we leverage the Gumbel softmax trick as (Muqeeth et al., 2023; Nie et al., 2021), where the router can be written as:

$$\hat{R}(x)_i = \frac{(log(R(x)_i) + g_i)/\tau}{\sum_{i=1}^{k} exp((log(R(x)_i) + g_i)/\tau)} \tag{7}$$

where $g_i \sim$ Gumbel$(0, 1)$ and τ is the temperature.

Due to MoE not being easily scalable and arbitrarily adding new LORAs, we randomly selected a LoRA as an expert for each task cluster in the experiment and trained the corresponding Router's parameters. We randomly selected 20 samples for each task during training to form a unified dataset for parameter training.

F.2 SMEAR

SMEAR (Muqeeth et al., 2023) does not perform routing aggregation on the Adapter output but rather aggregates the Adapter at the parameter level. We adopt the same setting as the MoE methods, and the results could be calculated in the following way:

$$\Theta_{SMEAR} = \sum_{i=i}^{k} R(x)_i \Theta_i, \tag{8}$$

where Θ_i denote the parameter of the LoRA-i.

F.3 AdapterSoup

AdapterSoup (Chronopoulou et al., 2023), for new downstream tasks, retrieves the parameters that need to be involved in aggregation through sentence bert and performs weight-space averaging on these parameters to adapt to the new domain. We have uniformly retrieved 3 LoRAs for mixed-task to test their capabilities under mixed-task conditions.

F.4 LoRAhub

LoRAhub (Huang et al., 2023) also aggregates 20 LoRAs randomly for new downstream tasks. In order to learn the weight of LoRA, a black-box optimization method is employed to learn the weight of each LoRA without calculating the gradients of the large model. It performs weighted averaging at the parameter level. Similar to the training process of MoE, we randomly selected 20 samples for each task to form a unified training dataset for black-box optimization.

G More Related Works

Personalized LoRA serving. Sheng et al. (2023) propose S-LoRA to discuss serving thousands of concurrent LoRA. The framework targets scenarios in which multiple tasks must be handled simultaneously without compromising the efficiency of the base models. Wen and Chaudhuri (2023) propose FLoRA, which enables efficient batching of diverse request types in low-rank adaptation (LoRA) of foundation models. These studies discuss how to deploy or train personalized LoRAs. However, these methods can only utilize a single user-specified LoRA during inference, failing to fully leverage the combination of LoRAs from different tasks. Moreover, the primary focus of these discussions is on computational strategies in GPUs and training strategies, which are orthogonal to the routing strategies with which we are concerned.