Convolutional LoRA Aggregation for Unseen Tasks Adaptation

Xinhao Wu, Jialin Liu, Yutai Duan, Jie Liu*
College of Artificial Intelligence, Nankai University, Tianjin 300350, CHINA
{2120240573,2120220573,ytduan}@mail.nankai.edu.cn
jliu@mail.nankai.edu.cn

Abstract

Recent studies have increasingly explored the combination of existing LoRA modules for effective adaptation to unseen tasks in data-scarce scenarios. However, current LoRA selection methods typically rely on a few task samples, making it difficult to capture the full scope of task-relevant information. Furthermore, even after selection, a knowledge gap remains between the selected LoRA modules and the target task, which existing coarse-grained LoRA aggregation strategies struggle to bridge. To address these challenges, we propose Selection and Convolution for LoRA aggregation (SC-LoRA), a two-stage framework that first selects appropriate LoRA modules based on parameter clustering and then aggregates them using a convolutional LoRA aggregator. Our LoRA selection strategy ensures comprehensive coverage of task-relevant LoRA modules by leveraging their distance in the parameter space. Building on this, the convolutional LoRA aggregator extracts useful knowledge in a fine-grained manner, seamlessly bridging the gap to the target task. Our experiments demonstrate that SC-LoRA excels in aggregating multiple LoRA modules for effective adaptation to unseen tasks.

1 Introduction

Fine-tuning with Low-Rank Adaptation (LoRA) (Hu et al., 2022) has become a prevalent paradigm for adapting large language models (LLMs) (Guo et al., 2025; Zhang and Shao, 2024) to downstream tasks. LoRA freezes the base model and trains lightweight parameter matrices, enabling task-specific knowledge to be encapsulated in a modular structure. Therefore, existing LoRA modules are naturally reusable for adapting to unseen tasks without requiring extensive labeled data.

Based on this, recent studies have explored how to combine existing LoRA modules to adapt to unseen tasks (Zhang et al., 2023; Chronopoulou et al., 2023; Wang et al., 2024a). However, existing approaches are still faced with significant challenges in selecting appropriate LoRA modules and aggregating them.

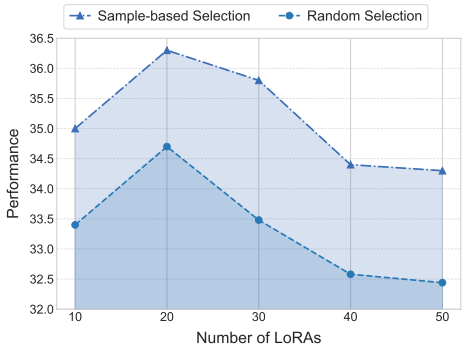

Figure 1: As the number of selected LoRA modules increase, the performance of Random Selection (LoraHub) and Sample-based Selection decreases. The Sample-based Selection builds on LoraHub and matches LoRA modules by computing the similarity between a few task samples and the training data of each module.

Among LoRA selection methods, random selection (Huang et al., 2024) often leads to the inclusion of LoRA modules that are irrelevant to the new task. Wang et al. (2024b) compute the similarity between the training data of each LoRA module and a few samples of the target task, which are insufficient to capture the full distribution of the new task. As a result, increasing the number of selected LoRA modules leads to a performance drop, as shown in Figure 1. Beyond the top-ranked LoRA modules that may capture the overall characteristics of the target task, additional ones are selected based on partial task information reflected by a few samples, which may resemble features encoded in irrelevant LoRA modules and lead to their incorrect selection, resulting in limited coverage of effective knowledge for the target task.

Crucially, after the LoRA modules are selected,

*Corresponding author

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 3702–3714
November 4-9, 2025

the combination process still suffers from a gap between the knowledge encapsulated in LoRA modules and the target task, since they are trained on upstream tasks that inherently differ from the unseen task. Most existing methods adopt coarse-grained composition strategies (Wang et al., 2024a; Huang et al., 2024; Wu et al., 2024) that operate at the level of entire parameter matrices or hidden vectors, failing to disentangle useful signals from redundant information within LoRA modules. Meanwhile, these strategies result in a relatively narrow range of representation space due to simplistic combination methods such as linear combination, making them inadequate for handling complex tasks. Therefore, if these LoRA modules are composed in a coarse-grained manner, the knowledge gap may be further exacerbated.

To address these challenges, we propose Selection and Convolution for LoRA aggregation (SC-LoRA), a novel framework that first selects appropriate LoRA modules based on parameter clustering and then integrates their representations in a fine-grained manner through a convolutional LoRA aggregator. Specifically, SC-LoRA clusters LoRA modules in the parameter space and identifies task-relevant clusters. This approach broadens the scope of useful knowledge by exploiting parameter-level correlations among LoRA modules, thereby overcoming the limited knowledge coverage inherent in sample-based selection methods. The convolutional LoRA aggregator employs convolutional kernels to scan the semantic feature map, which formed by concatenating the LoRA representations, enabling fine-grained extraction of task-relevant signals across LoRA modules while pruning redundant feature. Moreover, convolutional operation facilitates deep reconstruction of LoRA outputs, enhancing representational capacity and promoting tighter alignment with the target task to seamlessly bridge the knowledge gap. In summary, our main contributions are summarized as follows:

- We propose SC-LoRA, a novel framework for reusing knowledge from trained LoRA modules to tackle unseen tasks by identifying relevant LoRA modules through parameter-based clustering and composing them effectively.
- We design a convolutional LoRA aggregator that scans the semantic representations of selected LoRA modules, enabling fine-grained fusion of diverse knowledge, bridging the knowledge gap with the target task.
- Extensive experiments demonstrate that SC-LoRA consistently outperforms baselines in adapting multiple LoRA modules to unseen tasks under limited data conditions.

2 Related Work

The plug-and-play nature of LoRA modules has inspired extensive research to reuse multiple LoRA adapters for new tasks (Hu et al., 2024; Ouyang et al., 2025; Liu et al., 2024) from various perspectives.

LoRA Selection. Existing methods for selecting LoRA modules adopt several strategies. LoraHub (Huang et al., 2024) uses random sampling, while others (Wang et al., 2024b; Zhao et al., 2024) rely on task-specific samples to assess module adaptability to new tasks. Evaluation metrics include target loss (Huang et al., 2024), semantic similarity to source training data (Wang et al., 2024b), and matching probabilities from a trained embedding model (Zhao et al., 2024). However, in few-shot settings, these sample-based LoRA selection strategies depend on scarce task samples that fail to reflect the full task distribution, leading to inadequate knowledge coverage.

Combination of LoRAs. Another line of work explores the combination of multiple LoRA modules to enhance performance on new tasks(Zhong et al., 2024; Zou et al., 2025; Asadi et al., 2024; Tian et al., 2024).Meanwhile, numerous composition methods have also been proposed that build upon the idea of transferring knowledge from trained LoRA modules. LoRAFlow (Wang et al., 2024a) and LoRASoups (Prabhakar et al., 2024) enhance performance by aggregating domain-specific LoRA modules for decomposed subtasks. LoRAFlow employs a mixture-of-experts mechanism for token-wise weighting, while LoRASoups demonstrates that combining task-specific LoRAs outperforms single-module training on mixed-task data. MoLE (Wu et al., 2024) replaces MoE experts with source-specific LoRA modules, enabling dynamic and efficient composition without altering their individual characteristics. In addition, Arrow (Ostapenko et al., 2024) explores clustering and MoE-based techniques to build and reuse a LoRA library. These coarse-grained strategies often fail to preserve the unique properties of each LoRA module and may hinder effective knowledge integration.

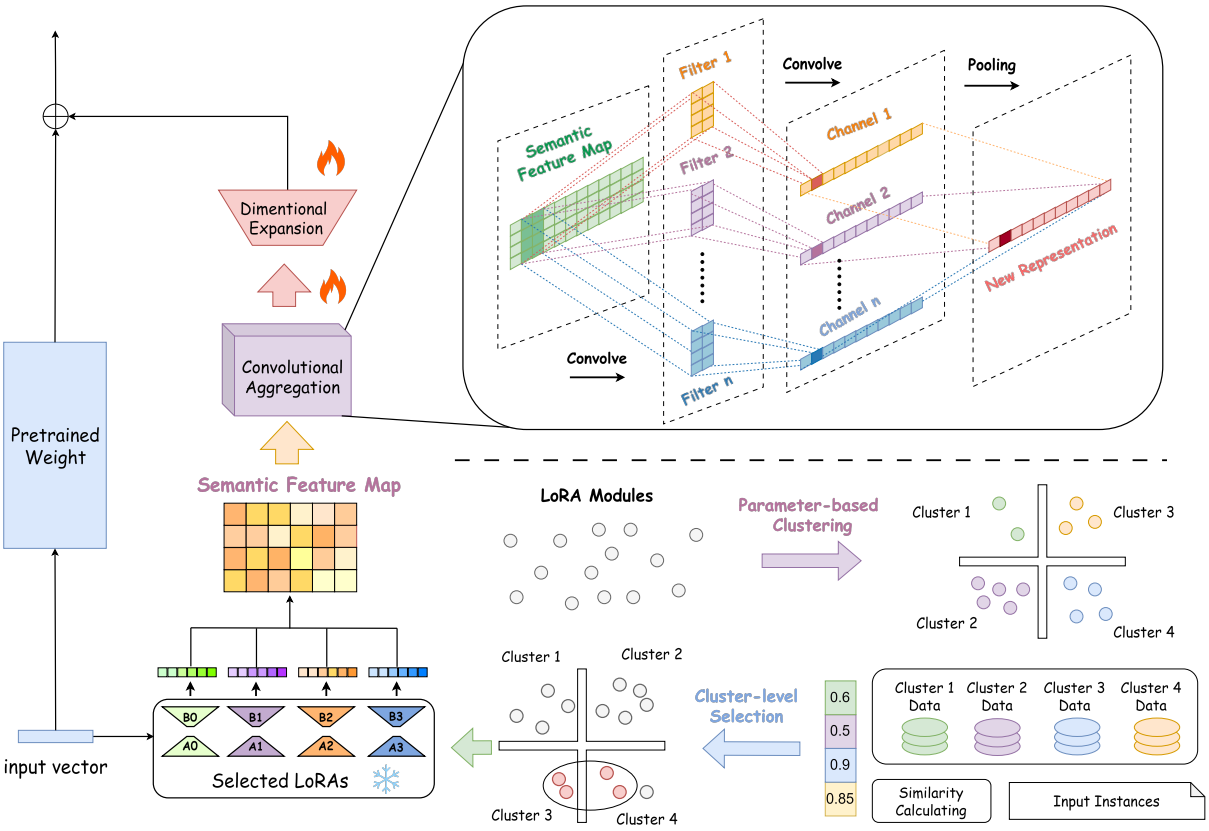

Figure 2: An overview of SC-LoRA. During training process, both the LoRA modules and the pretrained model weights are frozen, while the convolutional layer and subsequent linear layers remain trainable.

3 Preliminaries

LoRA LoRA(Hu et al., 2022) is a parameter-efficient fine-tuning technique designed to adapt LLMs to downstream tasks. Specifically, given the linear transformation matrix $W \in \mathbb{R}^{m\times d}$ in the language model $\mathcal{M}$, LoRA introduces a low-rank decomposition to approximate the parameter update to W, which is formulated as:

$$\Delta W = BA, \tag{1}$$

where $A \in \mathbb{R}^{r\times d}$ is initialized with Kaiming Uniform (He et al., 2015) and $B \in \mathbb{R}^{m\times d}$ initialized with zeroes. During fine-tuning, the original weight matrix W is kept frozen, and only the low-rank matrices A and B are updated.

Problem Statement Given a set of tasks $\mathcal{T} = \{T_1, T_2, ..., T_n\}$ and respective training sets $\mathcal{D} = \{D_1, D_2, ..., D_n\}$, the trained LoRA modules $\mathcal{L}$ are obtained by fine-tuning the language model $\mathcal{M}$ with LoRA on each task T_i using its corresponding training set D_i, for $i = 1, ..., n$.

We consider an unseen task $T \notin \mathcal{T}$, accompanied by only a few labeled examples t of this task. Given the inherent gap between training distributions $\mathcal{D}$ and the target task T, our goal is to explore how to use trained LoRA modules $\mathcal{L}$, in conjunction with the few-shot examples, to bridge the knowledge gap.

4 SC-LoRA

In this section, we elaborate the methodological details of SC-LoRA, which selects task-relevant LoRA modules by clustering them in the parameter space and employs CNN-based scanning to capture transferable knowledge in a fine-grained manner, as shown in Figure 2.

4.1 Cluster-level LoRA Selection

The trained LoRA modules are each specialized for its respective task and together provide a rich source of knowledge for addressing unseen tasks. Given the excessive size of the LoRA modules, it is more practical to select a subset of task-relevant LoRA modules rather than integrating all of them into a base model. As illustrated in Figure 1, sample-based selection suffers from performance degradation as the number of LoRA selected increases, suggesting that relying solely on limited

samples may be suboptimal for identifying LoRA modules well-suited for new tasks.

To this end, we propose a cluster-level LoRA selection (CLS) based on parameter clustering, as shown in the lower right of Figure 2, aiming to discover LoRA modules relevant to the target task but difficult to identify from limited sample imformation. CLS first performs parameter-based clustering to group LoRA modules into clusters and then selects clusters according to sentence similarity.

Formally, given trained LoRA modules $\mathcal{L} = \{L_1, L_2, ..., L_n\}$ and their corresponding training sets $D = \{D_1, D_2, ..., D_n\}$, we define the pairwise distance between two LoRA modules as:

$$d(L_i, L_j) = \sum_{k=1}^{m} \|W_{ik} - W_{jk}\|_F, \tag{2}$$

where m denotes the number of weight matrices in each LoRA module, and $\|\cdot\|_F$ denotes the Frobenius norm (Brauer and Shockley, 1962). We replace the standard Euclidean distance in k-means (MacQueen, 1967) with this LoRA-specific distance metric for clustering LoRA modules:

$$(O_1, O_2, ..., O_k) = \text{k-means}(\mathcal{L}, d), \tag{3}$$

where each cluster O_i $(i = 1, 2, ..., k)$ contains LoRA modules that are close in parameter space and similar in parameter structure.

To further guide LoRA cluster selection, we establish a metric to measure the correlation between the target task and each LoRA cluster. For a LoRA cluster O_i, let N denote the number of LoRA modules it contains, and let D_{ij} represent the training set of the j-th LoRA module in the cluster, where $j = 1, 2, ...N$.

We construct a representative sample set R_i for each cluster by sampling instances from each training set D_{ij}, so that R_i captures the distributed semantics of all LoRA modules in the cluster. Let t denote the examples of the target task. We compute the cluster-task relevance score as follows:

$$S_i = \sum_{t} \sum_{x \in R_i} \text{sim}(x, t), \tag{4}$$

where $\text{sim}(x, t)$ denotes the similarity between two samples, computed using an embedding model. We then prioritize clusters with higher relevance scores for knowledge transfer. More details of our LoRA selection method are provided in Appendix B.

CLS leverages the ability of k-means to maximize intra-cluster consistency, which helps ensure functional consistency among LoRA modules within each cluster. As a result, it reduces the risk of missing latent task-relevant LoRA modules and promotes broader LoRA selection, thereby extending the scope of transferable knowledge.

4.2 Convolutional LoRA Aggeragator

As the first step of SC-LoRA, CLS performs comprehensive selection of task-relevant LoRAs, providing rich transferable knowledge for new tasks. However, despite this improved alignment with the target task, the selected modules still exhibit a knowledge gap, which stems from intrinsic differences in the data distribution between their training tasks and the unseen task.

Moreover, in terms of composing LoRA modules, most existing methods adopt coarse-grained strategies, such as Mixture of Experts (MoE), which apply simple linear combinations of LoRA representations at a coarse granularity. Such aggregation fails to extract task-relevant signals from local representations across LoRA modules, may even undermine the knowledge encoded in the LoRA modules, and ultimately exacerbates the knowledge gap.

To address this issue, we draw inspiration from the concept of Convolutional Neural Networks (CNNs) (LeCun et al., 1998). CNNs are well-known for their ability to perceive local patterns and extract critical features. Adapting this property of CNNs to SC-LoRA, we apply convolutional kernels to scan LoRA outputs along the representation dimension, thereby capturing transferable knowledge across tasks in a more fine-grained manner, as illustrated in the upper right of Figure 2.

Formally, we concatenate the output representations of all LoRA modules into a matrix H, referred to as a semantic feature map, defined as:

$$H = \text{Concat}(h_1, h_2..., h_m), \tag{5}$$

where $H \in \mathbb{R}^{d \times m}$, and $h_i \in \mathbb{R}^d$ is the output of the i-th LoRA module, for $i = 1, ..., m$.

Let n 1D convolutional kernels be parameterized by $W_k \in \mathbb{R}^{d_w \times m \times n}$, where $d_w \times m$ defines the receptive field size of each kernel. Here, d_w is the window size along the representation dimension $(d_w < d)$, which determines the range of local representations captured from each module. We use a stride of 1 and no padding along the representation dimension. Applying the convolutional operation over the semantic feature map yields the fused representation, formulated as:

$$M = \text{Conv}(W_k, H), \tag{6}$$

Table 1: Performance comparison of SC-LoRA and other methods on the BBH. For each task, SC-LoRA uses 5 samples to identify suitable LoRA modules, and the same samples are used for subsequent fine-tuning with Flan-T5-Large as the base model.

Task	Zero	ICL	IA3	LoRA	FFT	LoRA MoE	LoRA MoE*	LoraHub	LoraHub*	IDLC	SC-LoRA
Boolean Expressions	54.0	59.6	56.2	56.0	62.2	53.3	56.7	55.5	54.6	49.8	61.3
Causal Judgement	57.5	59.4	60.2	55.6	57.5	60.9	59.8	54.3	59.7	59.9	54.0
Date Understanding	15.3	20.4	20.0	35.8	59.3	21.3	24.0	32.9	52.7	31.8	75.1
Disambiguation	0.0	69.1	0.0	68.0	68.2	0.0	0.0	45.2	63.3	46.9	71.3
Dyck Languages	1.3	0.9	4.2	22.2	19.5	0.7	1.3	1.0	1.3	0.0	12.2
Formal Fallacies	51.3	55.3	51.5	53.6	54.0	52.0	52.0	52.8	50.7	53.5	56.4
Geometric Shapes	6.7	19.6	14.7	24	31.1	16.7	18.7	7.4	6.7	18.8	35.1
Hyperbaton	6.7	71.8	49.3	55.3	77.3	69.3	71.3	62.8	54.7	71.8	76.7
Logical Deduction (five objects)	21.3	39.1	32.7	40.0	42.2	36.0	38.0	36.1	42.7	43.4	21.1
Logical Deduction (seven objects)	12.7	40.7	33.8	37.3	44.9	40.7	45.3	36.8	34.0	40.8	48.2
Logical Deduction (three objects)	0.0	51.6	8.5	53.6	52.9	27.3	32.7	45.7	50.7	51.0	51.3
Movie Recommendation	62.7	55.8	61.8	51.5	66.0	60.7	60.0	55.3	64.7	50.2	62.2
Multistep Arithmetic	0.7	0.7	0.7	0.2	0.0	0.7	0.7	0.4	0	0.4	1.3
Navigate	47.3	45.3	46.2	48.0	48.0	44.7	46.0	47.1	50.0	56.3	49.3
Object Counting	34.7	32.4	35.1	38.7	35.6	34.7	36.0	33.7	32.0	31.4	36.9
Penguins in a Table	43.5	41.3	45.0	36.2	31.9	43.5	41.3	35.9	43.5	36.9	38.4
Reasoning about Colored Objects	32.0	40.2	40.7	39.6	37.6	38.0	40.0	40.0	43.3	38.0	30.7
Ruin Names	23.3	19.3	24.4	37.8	61.3	22.7	24.7	24.4	21.3	22.0	37.1
Salient Translation Error Detection	37.3	47.3	37.1	16.0	16.2	40.7	47.3	36.0	37.3	31.0	44.4
Snarks	50.0	54.2	53.9	55.6	66.7	55.1	56.4	56.9	51.3	58.4	56.4
Sports Understanding	56.0	54.7	55.1	56.5	54.0	55.3	57.3	56.7	53.3	46.5	56.0
Temporal Sequences	16.7	25.1	18.2	25.1	37.8	18.0	18.7	18.2	17.3	24.9	46.7
Tracking Shuffled Objects (five)	12.0	12.0	12.0	13.8	16.9	12.0	12.0	12.3	12.0	11.0	23.8
Tracking Shuffled Objects (seven)	6.7	6.7	6.7	10.0	9.8	6.7	6.7	7.7	7.3	8.6	6.7
Tracking Shuffled Objects (three)	24.7	31.1	30.7	30.9	32.0	30.7	30.7	29.2	30.7	32.2	38.2
Web of Lies	54.0	53.8	54.2	52.7	48.2	54.0	54.7	50.1	53.3	44.1	48.7
Word Sorting	1.3	0.5	1.3	4.9	4.9	1.3	0.7	1.1	1.3	0.81	4.7
Avg Performance Per Task	27.0	37.3	31.6	37.7	42.1	33.2	34.5	34.7	36.7	35.6	**42.4**

where $M \in \mathbb{R}^{(d-d_w+1)\times n}$, with each column representing the output of a convolutional kernel scanning across the representation dimension.

we perform max pooling to reduce redundancy across the outputs of multiple channels, followed by a linear projection. The forward process after the convolution can be expressed as:

$$h = W_0 x + \text{MaxPool}(M) \cdot U, \tag{7}$$

where $W_0 \in \mathbb{R}^{d_{out}\times d}$ denotes the pretrained weight of the linear layer augmented with LoRAs, and $U \in \mathbb{R}^{(d-d_w+1)\times d}$ is a learnable projection matrix.

SC-LoRA employs trainable 1D convolutional kernels in a sliding window fashion to scan local representations across multiple LoRA modules along the hidden dimension. Compared to MoE, this design operates at a finer granularity across representations from multiple LoRA modules, enabling more localized operations.

Building upon this finer-grained operation, convolutional kernels learn to selectively preserve or suppress semantic information. This addresses the limitation of MoE methods, which operate only on entire representations and fail to capture transferable knowledge at a more granular level. As a result, SC-LoRA achieves a deeper integration of LoRA local representations from LoRA modules and overcomes the expressiveness bottleneck imposed by linear combinations.

In summary, the convolutional LoRA aggregator effectively integrates refined knowledge with enhanced expressive capacity, smoothly bridging the knowledge gap, and facilitating the transfer of multiple LoRA modules to unseen tasks.

5 Experiments

This section presents an overview of the SC-LoRA experiments, including the setup, implementation details, and analysis of its effectiveness.

Base Models and Benchmark. Our experiments are conducted using the publicly available LoRA modules released by LoraHub and LoRARetriever. To ensure compatibility and preserve the effectiveness of the LoRA modules, we reuse them to the same backbone models they were originally trained with: Flan-T5-Large (Chung et al., 2024) for LoraHub, and Llama-2-7B, 13B (Touvron et al., 2023) for LoRARetriever.

We follow the same benchmark and evaluation protocol as used in LoraHub (Huang et al., 2024), specifically evaluating on Big-Bench Hard (BBH) (Suzgun et al., 2022), a challenging benchmark for language model that encompasses 27 tasks from multiple domains. We report Exact Match

(EM) as the evaluation metric.

Implementation of SC-LoRA. To select relevant LoRA modules, SC-LoRA first applies k-means clustering to group LoRA modules into distinct clusters. For each cluster, we construct a representative sample set by randomly sampling from the training data of its LoRA modules. We then compute the similarity between each cluster and the target task using the bge-large-en model (Xiao et al., 2024). SC-LoRA incrementally selects LoRA modules from the top ranked clusters until a predefined module budget (10 by default) is met. If the number of LoRA modules in the selected clusters exceeds the budget, the last cluster is truncated; if it falls short, additional ones are sampled from the next-ranked clusters. Further details on the clustering and selection process can be found in the Appendix.B.

During the LoRA composition phase, SC-LoRA trains only the convolutional kernels and the up-projection matrix, keeping the base model and all trained LoRA modules frozen. When using Flan-T5-Large as the backbone, we set the convolutional kernel window size to 736 with 4 channels. For Llama-based backbones, the kernel window size is set to 2560 with 6 channels. SC-LoRA is trained using a small number of target task examples in a few-shot setting. Specifically, we use 5 examples and train for 10 epochs when using Flan-T5-Large as the backbone and 10 examples with 20 epochs when using the larger Llama-based models.

Baselines. • First, we compare SC-LoRA with the following methods that do not employ multiple LoRA modules: (1) Full Fine-tuning **(FFT)**; (2) LoRA tuning **(LoRA)** (Hu et al., 2022); (3) IA3 tuning **(IA3)** (Liu et al., 2022); (4) In-context Learning **(ICL)**; (5) Zero-shot Learning **(Zero)**. • Second, we extend the experiments to methods involving multiple LoRAs: (1) **LoRA MoE** (Liu et al., 2023; Zadouri et al., 2023; Wu et al., 2024) combines multiple LoRAs with a mixture of experts framework; (2) **LoraHub** (Huang et al., 2024) randomly selects 20 LoRA modules and aggregates them with gradient-free optimized weights for generalization to unseeen tasks; (3) Instance-Level Dynamic LoRAs Composition **(IDLC)** (Wang et al., 2024b) selects LoRA modules based on input instances, allowing different samples from the same task to utilize different combinations of LoRA modules; (4) **HydraLoRA** (Tian et al., 2024) introduces an architecture that shares a common matrix A across tasks while assigning each task a distinct matrix B, and employs a trainable MoE router to identify intrinsic components in the training data. (5) **LoraHub***, **LoRA MoE*** are variants of their base methods, both equipped with CLS (our LoRA selection strategy). Further details of the baseline methods are provided in Appendix A.

Table 2: Comparative performance of SC-LoRA, HydraLoRA, LoraHub tuning, Base model, mixture of lora experts (LoRA MoE), and LoRA tuning with the Llama-2-7b, Llama-2-13b as base model. For every task, SC-LoRA performs adaptation using 10 samples. Refer to Appendix C for detailed results.

Method	Llama2-7b	Llama2-13b
Base	31.6	38.4
LoRA	36.8	40.1
LoraHub	39.7	41.9
LoRA MoE	40.3	43.7
HydraLoRA	41.5	44.1
SC-LoRA	**42.8**	**47.2**

5.1 Main Result

As shown in Table 1, SC-LoRA significantly outperforms other baselines when using the Flan-T5-Large base model. This result indicates that SC-LoRA can effectively transfer knowledge from LoRA modules to complex target tasks by training on only a few examples.

One key factor behind this performance is our LoRA selection strategy, CLS. For example, LoRAHub* outperforms the original LoRAHub, illustrating that adaptive selection leads to better task alignment than random selection.

Another key factor lies in the convolutional LoRA Aggregator, which enables fine-grained local feature scanning and aggregation across the representations of different LoRA modules, facilitating a more task-aligned knowledge integration. Specifically, while both LoRA MoE* and IDLC employ strategically designed LoRA selection, they perform a coarse-grained composition on the selected LoRA modules. SC-LoRA achieves a 22.9% improvement over LoRA MoE* and a 16.0% improvement over IDLC, demonstrating the effectiveness and necessity of a finer-grained LoRA composition.

Notably, SC-LoRA slightly outperforms full fine-tuning (FFT), breaking the conventional trade-off between parameter efficiency and performance. On

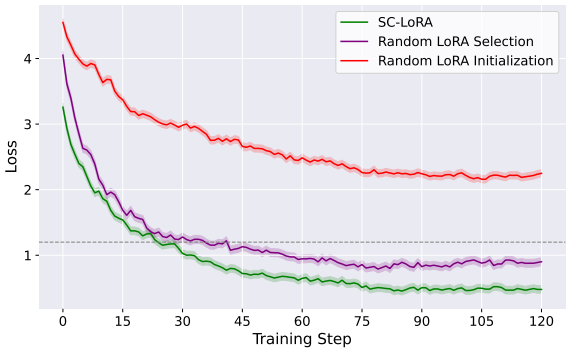

Figure 3: Training loss under different settings of LoRA modules, including SC-LoRA, random selection, and random initialization with Kaiming Uniform.

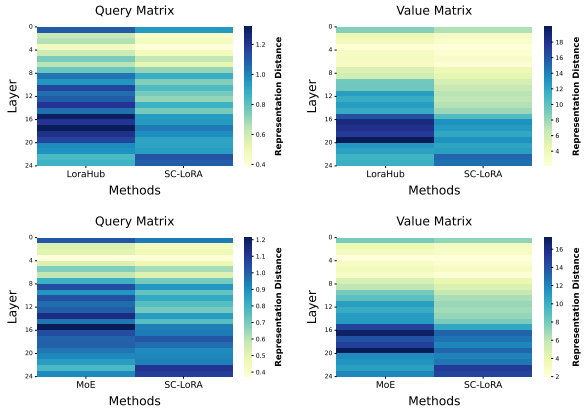

Figure 4: Representation Distance on target matrices (Q and V) Across different Methods.We compute the average representation distance across ten sample pairs from the date understanding task. In the visualization, lighter colors indicate shorter distances (i.e., more similar representations), while darker colors represent larger distances (i.e., more divergent representations).

the one hand, this benefit stems from the introduction of external knowledge from trained LoRA modules. The multi-domain external knowledge effectively compensates for the lack of sufficient data, helping the model achieve competitive results. On the other hand, it reflects SC-LoRA's strong ability to integrate external knowledge with task-specific examples, effectively bridging the gap between knowledge embedded in LoRA modules and the requirements of the target task.

As shown in Table 2, we explore the performance of SC-LoRA on larger models. From the results, SC-LoRA consistently outperforms other baselines on both the 7B and 13B model parameter scales, indicating its stable effectiveness across different model sizes. Moreover, compared to HydraLoRA, SC-LoRA does not rely on a large-scale data training process. Instead, it achieves superior performance by composing existing LoRA adapters and learning from a small number of samples. This not only highlights the simplicity of SC-LoRA's training process but also demonstrates its remarkable advantage in task adaptation.

5.2 Effectiveness Analysis

LoRA Selection Strategies. The LoRA selection strategy plays a critical role in adapting trained LoRA modules to unseen tasks. Selecting a subset of LoRA modules that closely matches the knowledge demands of the target task can significantly reduce the cost of knowledge transfer and promote more efficient task alignment.

As shown in Figure 3, we use randomly initialized LoRA modules as a baseline. The results show that this baseline struggles to converge during training, highlighting the necessity of transferring diverse knowledge from trained LoRA modules to the target task. Comparing the random selection strategy with SC-LoRA, random selection introduces a large amount of task-irrelevant knowledge in the early training steps, resulting in significantly higher loss and suboptimal final performance. In contrast, CLS selects a structurally relevant subset of LoRA modules with high transfer potential, leading to better convergence.

Representation Distance Visulization. We propose Representation Distance, a metric to evaluate a model's ability to learn generalizable features of the target task and fine-grained distinctions between similar samples. Taking the MoE-based model with LoRA as an example, we compute the distance between the outputs of two input samples at each MoE layer during the forward pass, which serves as the Representation Distance. We compute the Representation Distance of each LoRA augmentation module in MoE, LoraHub, and SC-LoRA, as shown in Figure 4.

Horizontally, we observe that across most layers, SC-LoRA exhibits significantly lower Representation Distance compared to MoE and LoraHub, indicating that the convolutional LoRA aggregator enables stable learning of task-consistent representations. This ability effectively brings the composed LoRA outputs closer to the task-specific representation space, thereby bridging the knowledge gap more smoothly.

Vertically, in the last few layers, SC-LoRA exhibits a clear increase in representation distance,

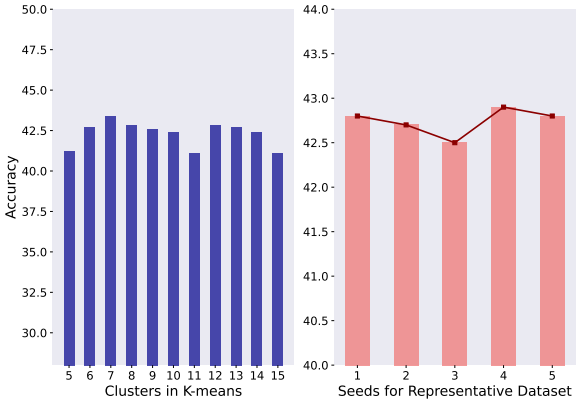

Figure 5: **(left)** Performance of SC-LoRA across different values of k in k-means, demonstrating robustness to the choice of cluster number. **(right)** The results remain stable with different random seeds when sampling representative datasets of LoRA clusters.

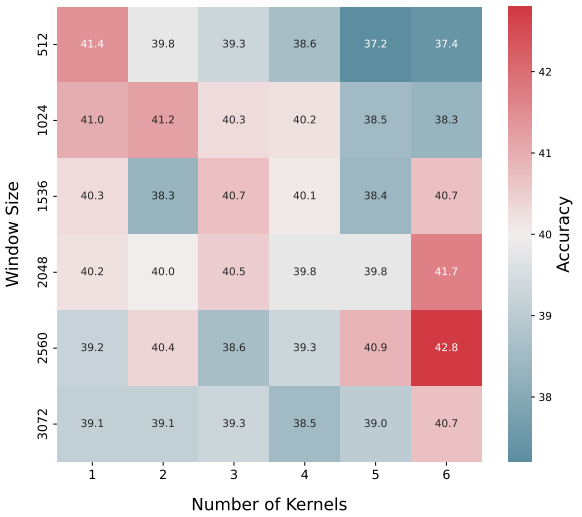

Figure 6: Performance of SC-LoRA vs. channels vs. kernel window size, with Llama-2-7b as the base model.

demonstrates stronger discriminative capability, and reveals finer semantic separation between samples. This indicates that convolutional aggregation enhances expressive capacity over linear combination, thus confirming the superior ability of SC-LoRA in feature composition and its support for a richer representational space.

5.3 Sensitivity Analysis

In this section, we analyze the parameter sensitivity of SC-LoRA. With all other configurations fixed, we vary the number of clusters in k-means, the selection of representative datasets, the number of convolutional channels, and the kernel sizes. We report the average accuracy of SC-LoRA in the 27 tasks in BBH (as reported in Table 1).

LoRA Selection. We design experiments on the hyperparameter settings of k-means and choice of representative datasets during the LoRA selection stage. Specifically, we vary the value of k and sample representative datasets for LoRA clusters with different random seeds, as shown in Figure 5. The results show that SC-LoRA is not sensitive to either the choice of k or the selection of representative datasets, suggesting that selecting an empirically reasonable k or representative dataset is sufficient for effective performance.

The robustness to k can be attributed to the diversity of the trained LoRA modules, which span a wide range of tasks. This diversity ensures that the LoRA modules are evenly distributed in the parameter space. Consequently, choosing k within a reasonable range allows k-means to consistently form well-balanced and semantically coherent clusters.

Convolutional Aggeragation. As shown in Figure 6, SC-LoRA maintains strong and stable performance in different configurations of channel and kernel window size. When the number of channels is small, smaller kernel sizes yield better results. As the number of channels increases, larger kernel sizes begin to perform better.

This observation reveals a collaboration between kernel size and channel count in SC-LoRA. Larger kernels cover broader semantic contexts and encode richer, more abstract information. In such cases, more channels are needed to decompose and interpret these complex patterns from multiple perspectives. In contrast, smaller kernels focus on fine-grained local features. When paired with fewer channels, they help avoid representational redundancy and allow the model to focus on constructing compact, high-quality local representations.

5.4 Ablation Studies

We conduct an ablation study focusing on the impact of the convolutional LoRA aggregator and the CLS LoRA selection mechanism. Figure 7 presents the results of our ablation study.

LoraHub serves as the baseline, performing random selection of LoRA modules followed by weighted combination. w/o CNN refers to a variant in which the convolutional LoRA aggregator is removed while the CLS mechanism is retained. w/o CLS refers to the variant in which the LoRA selection module is ablated, retaining only the con-

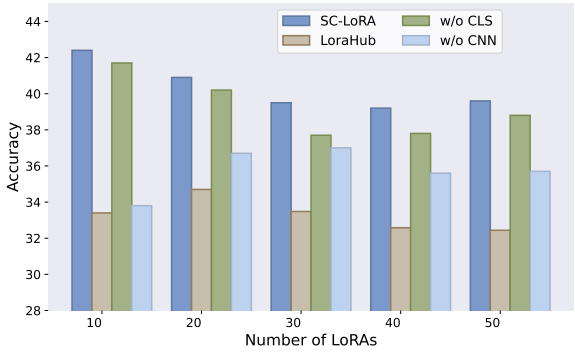

Figure 7: Comparative performance of ablation study for SC-LoRA on the BBH Benchmark, with Flan-T5-Large as the backbone.

volutional LoRA aggregator. Figure 7 shows that the full SC-LoRA model outperforms all ablated variants, demonstrating that both the CLS mechanism and the convolutional LoRA aggregator are essential to its effectiveness across diverse tasks.

6 Conclusion

In this paper, we propose SC-LoRA, a novel framework that enables leveraging the knowledge of trained LoRA modules to tackle unseen tasks under data-scarce conditions. SC-LoRA first performs parameter-based clustering and selects LoRA modules at the cluster level, overcoming the limited coverage of knowledge in sample-based selection. It then employs a convolutional aggregator to integrate the representations of the selected LoRA modules in a fine-grained manner, seamlessly bridging the knowledge gap between existing knowledge and unseen tasks. Experiments demonstrate that SC-LoRA consistently enables robust knowledge transfer across diverse tasks.

Limitations

In this section, we discuss the potential limitations of SC-LoRA. Our work focuses on trained LoRA modules transfer using LoRA, one of the most widely adopted parameter-efficient fine-tuning (PEFT) methods. However, we have not yet explored the applicability of SC-LoRA to other types of adapters, such as prompt tuning or prefix tuning. We leave the extension of SC-LoRA to other adapter types as a direction for future work. SC-LoRA partially leverages the original training data of each LoRA module to estimate its adaptability to new tasks during the LoRA selection stage. In privacy-sensitive conditions, where access to LoRA training data is restricted or prohibited, the proposed approach may not be applicable. Future work could explore privacy-preserving strategies for selecting LoRA modules without direct access to their training data.

Acknowledgments

This research is supported by the National Key Research and Development Program of China under the grant No. 2023YFF0725003, National Natural Science Foundation of China under the grant No. 62376129, Tianjin Science and Technology Major Project under the grant No. 24ZXZSSS00420, Tianjin Natural Science Foundation under the grant No. 24JCYBJC01950, and Tiankai Higher Education Science and Technology Park Enterprise R&D Special Project under the grant No. 23YFZXYC00029.

References

Nader Asadi, Mahdi Beitollahi, Yasser Khalil, Yinchuan Li, Guojun Zhang, and Xi Chen. 2024. Does combining parameter-efficient modules improve few-shot transfer accuracy? *arXiv preprint arXiv:2402.15414*.

Alfred Brauer and James E Shockley. 1962. On a problem of frobenius. *J. reine angew. Math*, 211(19):215–220.

Alexandra Chronopoulou, Jonas Pfeiffer, Joshua Maynez, Xinyi Wang, Sebastian Ruder, and Priyanka Agrawal. 2023. Language and task arithmetic with parameter-efficient layers for zero-shot summarization. *arXiv preprint arXiv:2311.09344*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, and 1 others. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, and 1 others. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Zixuan Hu, Yongxian Wei, Li Shen, Chun Yuan, and Dacheng Tao. 2024. Unlocking tuning-free few-shot adaptability in visual foundation models by recycling pre-tuned loras. *arXiv preprint arXiv:2412.02220*.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2024. Lorahub: Efficient cross-task generalization via dynamic loRA composition. In *First Conference on Language Modeling*.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

Jialin Liu, Jianhua Wu, Jie Liu, and Yutai Duan. 2024. Learning attentional mixture of loras for language model continual learning. *arXiv preprint arXiv:2409.19611*.

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

James MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, volume 5, pages 281–298. University of California press.

Oleksiy Ostapenko, Zhan Su, Edoardo Maria Ponti, Laurent Charlin, Nicolas Le Roux, Matheus Pereira, Lucas Caccia, and Alessandro Sordoni. 2024. Towards modular llms by building and reusing a library of loras. *arXiv preprint arXiv:2405.11157*.

Ziheng Ouyang, Zhen Li, and Qibin Hou. 2025. K-lora: Unlocking training-free fusion of any subject and style loras. *arXiv preprint arXiv:2502.18461*.

Akshara Prabhakar, Yuanzhi Li, Karthik Narasimhan, Sham Kakade, Eran Malach, and Samy Jelassi. 2024. Lora soups: Merging loras for practical skill composition tasks. *arXiv preprint arXiv:2410.13025*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, and 1 others. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Cheng-Zhong Xu. 2024. Hydralora: An asymmetric lora architecture for efficient fine-tuning. *Advances in Neural Information Processing Systems*, 37:9565–9584.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hanqing Wang, Bowen Ping, Shuo Wang, Xu Han, Yun Chen, Zhiyuan Liu, and Maosong Sun. 2024a. Lora-flow: Dynamic lora fusion for large language models in generative tasks. *arXiv preprint arXiv:2402.11455*.

Zhiqi Wang, Shizhu He, Kang Liu, and Jun Zhao. 2024b. Instance-level dynamic loras composition for cross-task generalization. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 5699–5708.

Xun Wu, Shaohan Huang, and Furu Wei. 2024. Mixture of lora experts. *arXiv preprint arXiv:2404.13628*.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 641–649.

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*.

Hong Zhang and Haijian Shao. 2024. Exploring the latest applications of openai and chatgpt: An in-depth survey. *CMES-Computer Modeling in Engineering & Sciences*, 138(3).

Jinghan Zhang, Junteng Liu, Junxian He, and 1 others. 2023. Composing parameter-efficient modules with arithmetic operation. *Advances in Neural Information Processing Systems*, 36:12589–12610.

Ziyu Zhao, Leilei Gan, Guoyin Wang, Wangchunshu Zhou, Hongxia Yang, Kun Kuang, and Fei Wu. 2024. Loraretriever: Input-aware lora retrieval and composition for mixed tasks in the wild. *arXiv preprint arXiv:2402.09997*.

Ming Zhong, Yelong Shen, Shuohang Wang, Yadong Lu, Yizhu Jiao, Siru Ouyang, Donghan Yu, Jiawei Han, and Weizhu Chen. 2024. Multi-lora composition for image generation. *arXiv preprint arXiv:2402.16843*.

Xiandong Zou, Mingzhu Shen, Christos-Savvas Bouganis, and Yiren Zhao. 2025. Cached multi-lora composition for multi-concept image generation. *arXiv preprint arXiv:2502.04923*.

A Details of Baselines

Methods without the Use of Multiple LoRAs

- **Full Fine-tuning** is a widely used stategy, often considered the upper bound of performance in adaptation. It updates all parameters of the backbone model by gradient optimization.
- **IA3** (Liu et al., 2022) is a lightweight fine-tuning method that injects learnable learned vectors into transformer structures.
- **ICL** (Dong et al., 2022) performs inference by including a few examples in the input prompt.
- **Zero** directly prompts the model without using any task-specific examples.

Methods Involving Multiple LoRAs

HydraLoRA. HydraLoRA first partitions the dataset into n domain clusters via k-means, then initializes a collection of n experts, denoted as $E_1, ..., E_n$.These experts serve a role analogous to the LoRA B matrices, sharing a central parameter matrix matrix A, and are orchestrated by a MoE router.The HydraLoRA architecture can be formulated as:

$$y = \sum_{i=1}^{n} w_i E_i A x, \tag{8}$$

where w_i is calculated from a trainable weight matrix $W \in \mathbb{R}^{d \times n}$ followed by a softmax function, which transforms the input representation x into scores $(w_1, ..., w_n)$ for experts:

$$w_i = \text{softmax}(W^T x). \tag{9}$$

HydraLoRA was trained on the Flanv2 datasets, which include Natural Language Understanding (NLU) and Natural Language Generation (NLG), and then evaluated on the BBH dataset using 3-shot learning.

LoraHub. Given m LoRA modulesL_i, for $i = 1, ..., m$, each module is represented as $L_i = A_i B_i$. LoraHub performs a weighted operation on the LoRA parameter matrices A_i and B_i, aggregating the parameters of multiple LoRA experts into a single parameter structure ΔW. The computation process is as follows:

$$\Delta W = (\sum_{i=1}^{m} w_i A_i)(\sum_{i=1}^{m} w_i B_i), \tag{10}$$

w_i represents the weight of each LoRA module, determined by a gradient-free algorithm that combines cross-entropy loss with L_1 regularization. For both backbones, we randomly select 20 LoRA modules. The training set consists of 5 samples when using the Flan-T5-Large, while 20 samples for LLaMA-2-7B, 13B.

B Details of LoRA Module Selection

B.1 K-means Clustering Settings

In this work, we address the challenge that many task-relevant LoRA modules are difficult to capture solely based on sample information. To overcome this limitation and expand the coverage of effectively transferred knowledge, we propose clustering the LoRAs using the k-means algorithm (MacQueen, 1967) for selection. Specifically, k-means is a widely used unsupervised machine learning algorithm that partitions a dataset, in our case a set of LoRA modules, into a pre-specified number of clusters based on feature similarities.

In our approach, these feature similarities are represented by the sum of the distances between all parameter matrices of LoRA modules. To prevent significant size disparities between clusters, we set an upper capacity limit t for each cluster and apply a recursive k-means procedure. In each iteration, k clusters are created, and any cluster exceeding the capacity limit is recursively split until the number of modules in each cluster is less than t. In our experiments, we set $k = 3$ and $t = 5$ for LLaMA, while for T5, we set $k = 10$ and $t = 30$, as the number of LoRA modules for T5 (from LoraHub) is considerably larger than for LLaMA (from LoraRetriever).

B.2 Visualization of LoRA Selection Results

SC-LoRA comprehensively selects LoRA modules aligned with the target task, thus taking the first step toward bridging the knowledge gap. Figure 8 illustrates the case of LoRA selection stage, using an example from the Salient Translation Error Detection task in BBH benchmark. The number of LoRA experts selected is set to 10. After the K-Means clustering and selection based on sentences similarity, most of the LoRAs retrievered by SC-LoRA are related to translation error correction, including four language task LoRA modules and two judgment task LoRA modules, which closely align with the task requirements. This demonstrates that SC-LoRA can effectively retrieve LoRA modules

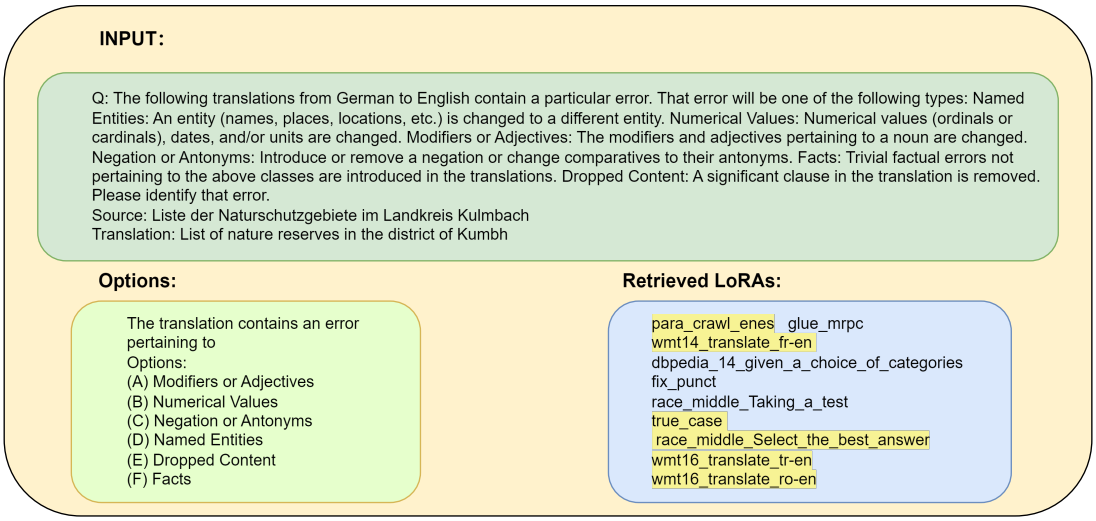

Figure 8: Visualization of SC-LoRA's selection of task-suited LoRAs.

relevant to the target task, providing a more robust knowledge foundation for the knowledge transfer of LoRA modules.

C More Results

Tabel 3 presents a comparative performance of different methods on the BBH benchmark, using LLaMA2-7B as the backbone model. The methods contain base model (Base), LoRA tuning (LoRA), LoraHub learning, multi-LoRA integrated with MoE (LoRA MoE), HydraLoRA learning and our proposed SC-LoRA. For SC-LoRA, 10 samples are used for LoRA selection and training processes.

Table 3: Comparative performance of baselines and SC-LoRA on the BBH benchmark using LLaMA-2-7B as the base models.

Task	**Base**	**LoRA**	**LoRAHub**	**MoE**	**HydraLoRA**	**SC-LoRA**
Boolean Expressions	61.9	67.1	72.9	68.0	73.7	67.3
Causal Judgement	52.2	54.9	50.1	51.4	53.2	47.1
Date Understanding	30.4	35.2	36.0	33.9	36.0	51.3
Disambiguation	34.8	45.2	42.1	47.2	50.3	62.7
Dyck Languages	15.8	18.7	14.5	16.8	19.8	48.0
Formal Fallacies	49.0	62.2	64.5	67.6	65.3	44.0
Geometric Shapes	9.7	17.7	18.7	17.7	19.7	44.7
Hyperbaton	51.8	74.3	74.3	68.9	77.2	62.0
Logical Deduction (five objects)	21.9	33.3	38.7	40.0	42.2	28.0
Logical Deduction (seven objects)	15.0	36.4	37.3	40.7	40.7	17.3
Logical Deduction (three objects)	32.8	41.4	38.5	43.7	42.9	38.7
Movie Recommendation	34.4	53.5	56.0	56.8	58.3	82.7
Multistep Arithmetic	1.2	1.2	1.9	1.9	1.8	1.3
Navigate	53.8	52.7	56.2	58.0	57.1	51.3
Object Counting	40.1	40.5	42.3	44.7	42.3	50.7
Penguins in a Table	21.7	23.2	25.0	23.2	25.9	26.1
Reasoning about Colored Objects	19.4	28.0	32.7	38.3	38.3	24.7
Ruin Names	24.3	28.7	34.3	34.3	36.7	24.7
Salient Translation Error Detection	11.3	11.1	17.1	16.2	20.1	16.7
Snarks	44.0	47.9	54.9	53.6	56.9	59.0
Sports Understanding	57.5	59.0	61.2	59.0	60.2	71.3
Temporal Sequences	21.1	32.6	28.9	34.1	30.4	89.3
Tracking Shuffled Objects (five objects)	21.9	23.7	23.7	28.0	29.3	21.3
Tracking Shuffled Objects (seven objects)	14.6	15.3	16.6	15.3	15.3	14.7
Tracking Shuffled Objects (three objects)	32.4	38.4	39.0	38.4	40.7	32.7
Web of Lies	51.4	52.8	53.2	50.1	52.0	47.3
Word Sorting	29.6	33.6	33.6	31.2	34.0	30.0
Avg Performance	31.6	36.8	39.7	40.3	41.5	42.8