Instance-Level Dynamic LoRAs Composition for Cross-Task Generalization

Zhiqi Wang[1,2] and **Shizhu He**[1,2*] and **Kang Liu**[1,2] and **Jun Zhao**[1,2]
[1] The Key Laboratory of Cognition and Decision Intelligence for Complex Systems,
Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
`wangzhiqi2022@ia.ac.cn`, {shizhu.he, kliu, jzhao}@nlpr.ia.ac.cn

Abstract

Large language models perform well on tasks that have undergone fine-tuning of instructions, but their performance on completely unseen tasks is often less than ideal. To overcome the challenge of cross-task generalization, **task-level LoRAs combination** is proposed, which does not require training a model for new tasks. Instead, it learns the LoRA modules combination weights based on a small number of samples to form the task model. However, task-level LoRAs combination only utilizes a few task modules due to its reliance on the weight enumeration method, and it also ignores the specificity between different instances. Therefore, we proposed an **instance-level LoRAs composition** for cross-task generalization, which selects appropriate multiple task LoRA modules for each input instance and dynamically determines the composition weights. Our experiments on publicly available datasets show that our method outperforms the typical method, LoraHub, in 16 out of 27 tasks. We release the source code at `https://github.com/noname822/iLoraComp.git`

1 Introduction

Currently, large language models (LLMs) demonstrate remarkable zero-shot learning capabilities on tasks that have undergone instruction tuning (Chung et al., 2022; Achiam et al., 2023; Touvron et al., 2023; AI@Meta, 2024). However, numerous studies have revealed that when encountering novel tasks outside their training distribution, these models often fail to exhibit satisfactory performance (Ovadia et al., 2024; Huang et al., 2024). Exploring strategies to enhance the cross-task generalization abilities of these massive language models, enabling them to adapt swiftly and accurately to diverse new tasks, has emerged as a pressing challenge that demands attention.

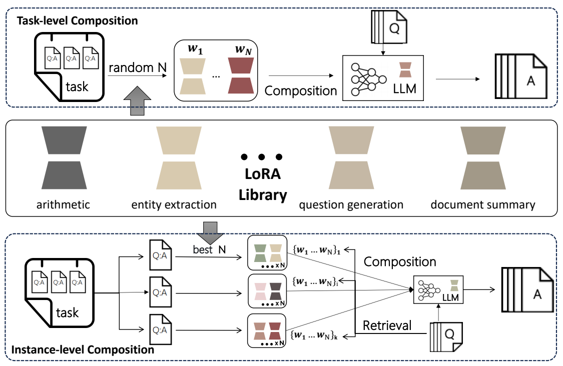

Figure 1: Previous task-level composition constructs a shared task model for all instances. The proposed instance-level composition constructs a unique task module for each instance.

Addressing the challenge of cross-task generalization has traditionally involved fine-tuning models for each task and in-context learning. However, these conventional approaches come with inherent limitations. Fine-tuning for every new task can be resource-intensive, demanding extensive data, storage, and computing power, which compromises flexibility. Although methods such as LoRA (Hu et al., 2021), falling under the delta tuning (Ding et al., 2022) approach, aim to adapt to specific tasks or domains by introducing smaller parameter updates while minimizing computation and storage costs, thus mitigating storage issues and enhancing flexibility, they still require backpropagation for precise output tuning, rendering them less cost-effective for multiple tasks. In-context learning (Dong et al., 2022), on the other hand, necessitates more input than zero-shot to fully leverage the model's capabilities, indirectly increasing the computational resources needed for inference.

To address the shortcomings of these methods and achieve efficiency and sustainability in multi-task, few-shot, and high-volume scenarios, innovative approaches such as LoraHub (Huang et al., 2024) have emerged. LoraHub rapidly adapts to

[*]Corresponding Author

unseen tasks by intelligently combining pre-trained low-rank adapters from other relevant tasks. This method enhances model performance across diverse tasks without increasing input requirements, striking a balance between performance and energy consumption.

However, LoraHub also has room for improvement in terms of its effectiveness. Firstly, when selecting LoRA modules from a trained LoRA library for task adaptation composition, LoraHub's current strategy is to randomly select modules from the library. This random selection may result in the inclusion of tasks that are either overly similar or completely unrelated, leading to significant performance variations under different random seeds for the same task, thus exhibiting poor stability. Secondly, when training on instances, LoraHub does not consider the subtle nuances between individual instances, preventing the full utilization of the limited instance data to capture the potential specificity of inputs, which in turn limits LoraHub's performance. To address these two issues, we propose the following solutions:

- To address the issue with the LoRA module selection strategy, we adopt a selection method based on task similarity. By calculating the semantic similarity between the target task and the training sets of the available LoRA modules, we prioritize the combination of LoRA modules that are most closely related to the current task, thereby enhancing the stability and effectiveness of the task-level adaptation.

- To fully account for the unique characteristics of each input instance, we propose tailoring a dedicated LoRA module combination for each instance. By calculating the semantic similarity between the input instance and the training instances used to create the available LoRA modules, we select the most fitting instance-specific LoRA combination as the processing strategy for that input. This approach effectively leverages the subtle nuances across different input instances.

By employing the aforementioned improvements, our method has achieved a significant enhancement in inference stability. Additionally, compared to the original LoraHub, our approach has demonstrated a noticeable performance advantage. In our experiments, a total of 27 tasks were tested, and in these, our proposed method outperformed LoraHub on 16 of them.

2 Related work

Instance-Based Generation for LLMs refers to a method that leverages dataset analysis to extract valuable instance, thereby enhancing the performance of a task. The introduction of large language models has since inspired numerous works (Xu et al., 2024), including Wiki-Chat (Semnani et al., 2023), EPR (Rubin et al., 2022), LLM-R (Wang et al., 2024b), which have sought to augment language model capabilities through retrieval-based knowledge enhancement. This trend originated with RAG (Lewis et al., 2020), which incorporates knowledge as prompts for in-context learning in LLM. Additionally, there are works that do not retrieve text as prompts, but instead retrieve delta-tuning modules, using these modules to generate prompts for answering questions, such as Knowledge Card (Feng et al., 2023). In this paper, we retrieve delta-tuning modules by calculating the semantic similarity between instance and question using the method of DPR (Karpukhin et al., 2020a).
Module Composition represents an endeavor to integrate diverse models, Consequently, tasks that retrieve model modules for composition have naturally emerged, such as MAC (Tack et al., 2024), SLM (Peng et al., 2024), Arrow (Ostapenko et al., 2024), LoraRetriever (Zhao et al., 2024), and LoraFlow (Wang et al., 2024a). While most methods adopt a simplistic processing approach for models. These approaches strive to leverage retrieval methods by employing retrieval scores as weights during composition, thereby obviating the need for manual parameter tuning and facilitating immediate usage. Concurrently, methods such as Moelora (Liu et al., 2023) exist that directly assign weights through backpropagation. LoraHub occupies an intermediary position that uses a gradient-free optimization. In comparison to previous work, our approach places a stronger emphasis on utilizing instances to get model modules that are more relevant to the given question.

3 Background

3.1 LoRA

LoRA (Hu et al., 2021) is an efficient parameter fine-tuning technique designed to address the issue of excessive storage space needed for model modifications. Specifically, in each selected linear layer, LoRA uses two low-rank matrices, $A \in \mathbb{R}^{r\times d}$ and $B \in \mathbb{R}^{m\times r}$. Here, $r \in \mathbb{Z}$ denotes the custom rank of the matrices, while $d, m \in \mathbb{Z}$ represent the original dimensions of the linear layer. These matrices are applied in Equation 1 to calculate the weight change matrix $\Delta W \in \mathbb{R}^{m\times d}$, which is subsequently incorporated into the linear layer during model inference.

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

3.2 LoraHub

LoraHub (Huang et al., 2024) is a method for adaptive integration of LoRA modules. With LoraHub, we can swiftly adapt to various tasks using a minimal number of instances, employing a gradient-free method (Sun et al., 2022). Through LoraHub, we obtain a set of weights $\hat{w} = \{w_1, \ldots, w_N\}$ for N LoRA modules. For every linear layer within LoRAs, the weights are combined according to Equation 2.

$$\Delta W_{combine} = \sum_{i=1}^{N} w_i \Delta W_i \tag{2}$$

4 Method

In this section, we will provide an overview of the process, followed by an explanation of how to identify appropriate task LoRA modules based on LoRA training data. Finally, we will offer a detailed account of how to integrate the selected LoRA combinations with the input data.

4.1 Overview

We first train the upstream tasks $\mathbb{T}$ on the large model M_θ using the training set $\mathcal{T}i \in \mathbb{T}$ to get LoRA module L_i and collect them into LoRA library $\mathbb{L}$. Next, We specify the hyperparameter N as the number of LoRA modules to be composed. Each new task $\mathcal{T}' \notin \mathbb{T}$ has their instance set $\mathcal{I}'$. For each instance $e_j \in \mathcal{I}'$, we find the closest N LoRA library from $\mathbb{L}$, denoted as $\mathcal{L}_{e_j} = \{L_1, \ldots, L_N\}$, and optimize a weight combination $\hat{w}_{e_j} = \{w_1, \ldots, w_N\}$ using a gradient-free method (Sun et al., 2022) as ng. For a new question Q belonging to new task T', we select the most suitable weight combination $\hat{w}_{e_j}$ based on the semantic similarity between Q and e_j then make new LoRA module $\hat{L}_j$. Finally, we combine these to form the model $M_\phi = LoRA(M_\theta, \hat{L})$ and use it for reasoning on Q.

4.2 LoRA module Retrieval

To select the most suitable LoRA modules from $\mathbb{L}$ for composition, we identify the corresponding training set $\mathcal{T}_i = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ for each $L_i \in \mathcal{L}$. We then derive the task embedding vector $emb_{\mathcal{T}i} = \frac{1}{n}\sum_{k=1}^{n} M_s(x_k + y_k)$ using the sentence vectorization model M_s. Similarly, for the instance $e_j = (x_{e_j}, y_{e_j})$, we can obtain its embedding vector $emb_{e_j} = M_s(x_{e_j} + y_{e_j})$. Consequently, Following the approach of Mussmann and Ermon, 2016 and Karpukhin et al., 2020b in using cosine similarity as a measure of task similarity, we can identify the top N most similar tasks to e_j. The formula for cosine similarity is as follows:

$$similarity(e_j, \mathcal{T}i) = \frac{emb_{e_j} \cdot emb_{\mathcal{T}i}}{\|emb_{e_j}\| \cdot \|emb_{\mathcal{T}i}\|} \tag{3}$$

Where $emb_{\mathcal{T}_i}$ represents the embedding vector of the i-th task, and $\|\cdot\|$ denotes the Euclidean norm of a vector. By calculating the cosine similarity between each task $\mathcal{T}i$ and the instance e_j, we can select the top N tasks with the highest similarity as the candidate set of similar tasks for e_j, which is denoted as $\mathcal{L}_{e_j}$, and then collect all $\mathcal{L}_{e_j}$ as a set called $S_{\mathcal{L}}$.

4.3 Instruct based Module Composition and Inference

To fine-tune the model M_θ to the state that best aligns with the instance $e_j = (x_j, y_j)$, we employ the non-gradient optimization method ng to refine the weights. We perform a broad adjustment of the init weights w_{init} using all the instances for $\mathcal{T}i$ donated as $\mathcal{I}_i = \{e_1, \ldots, e_n\}$. Then, we conduct a targeted adjustment using the instruct-level LoRA set $\mathcal{L}_{e_j}$ corresponding to the specific instance e_j. The optimization process is encapsulated in the following formula:

$$\hat{w}_{e_j} = ng(\mathcal{I}_i, \mathcal{L}_{e_j}, w_{init}) \tag{4}$$

Having aggregated the adjusted weights $\hat{w}_{e_j}$ for all e into the set $S_{\hat{w}}$, we proceed to identify the e_j that shares the most affinity with the input x. This is accomplished by calculating the cosine similarity between the input embedding vector $emb_{e_{ix}} = M_s(x_j)$ for e_j and the embedding vector $emb_x = M_s(x)$ for the input x. This analysis allows us to select the most suitable LoRA module from $S_{\mathcal{L}}$, denoted as $\mathcal{L}_{suit}$, and its corresponding weights from $S_{\hat{w}}$, denoted as $\hat{w}_{suit}$. Utilizing these components, we construct the optimal LoRA module $\hat{L} = \hat{w}_{suit}\mathcal{L}_{suit}$. As a result, we obtain the model $M_\phi = LoRA(M_\theta, \hat{L})$ that is specifically tailored to the given input. This model is then employed for inference, with the output expressed as $y = M_\phi(x)$.

5 Experimental Setup

LLM. We utilized the Flan-T5-Large (Chung et al., 2022) model as our foundational large language model M_θ for experimentation purposes. Concurrently, we employed the compact all_datasets_v4_MiniLM-L6 (flax sentence embeddings, 2021; Wang et al., 2020) model as our M_s, which was trained on a dataset comprising one billion sentence pairs, excluding the BBH and flanv2 datasets that we utilized. This compact model effectively supported our sentence vectorization efforts.

Dataset and Evaluation. We utilize the flanv2 dataset (Longpre et al., 2023), which incorporates data from four mixed sources, as the training set for upstream tasks. It encompasses 264 distinct datasets, out of which we selected 97 for our purposes. We then employed the LoRA modules trained on these datasets by Huang et al. (2024) as our repository of LoRA modules for potential selection.

The Big-Bench Hard benchmark (Suzgun et al., 2022), with 27 tasks, offers a valid test for M_θ as it was not trained on these datasets. We sampled 5 instances per task, used 20 LoRA modules for adaptation, and initiated with 40 steps of global optimization, followed by EM-based evaluation on the remaining data.

Baseline Setup. To ensure our method's credibility, we used our LoRA library to test LoraHub (Huang et al., 2024) refined parameters for 40 steps as a baseline, averaging three runs for the final score (LoraHub$_{\text{avg}}$). We compared scores using zero-shot, full fine-tuning (FFT), and in-context learning (ICL). For LoRA module selection, we conducted ablation experiments using the average embedding vector of five instances per task (BatchComp). In FFT, we maintained consistency by training with the same random seeds and 5 instances. We trained the model over 40 epochs with a learning rate of 3e-5 and batch size of 5.

Method	average	average-3
FFT*	<u>39.8</u>	<u>44.3</u>
0-shot	24.4	27.4
ICL	30.9	34.8
LoraHub$_{avg}$	34.0	38.1
BatchComp	34.7	39.0
Ours	**35.6**	**40.0**

Table 1: Experimental results on 27 tasks of BBH, the "average-3" has excluded three tasks with an accuracy of less than 10%, (*) represents the upper limit.

Method	FFT	ICL	0-shot	LoraHub
BatchComp	7/18	18/3	16/8	13/12
Ours	11/16	19/2	18/7	16/8

Table 2: In the A/B comparison across all 27 tasks, 'A' represents the number of tasks where our method outperformed the baseline, while 'B' represents the number of tasks where it underperformed compared to the baseline.

6 Result And Discussion

6.1 Result

The primary results are presented in Table 1 and Table 2, with detailed task scores in Appendix A. Our method significantly outperforms the zero-shot approach on 19 out of 27 tasks and the in-context learning (ICL) method on 18 tasks in terms of average performance. Compared to ICL, our approach is more computationally efficient, requiring fewer tokens. Our modifications to LoraHub are also notably successful, with our method outperforming LoraHub's random selection approach on 16 tasks. Crucially, our instance-level method exhibits a 0.9% performance enhancement over our task-level method in the ablation study, underscoring the efficacy of capturing input nuances through instance-specific adaptation.

However, our method still cannot compete with full fine-tuning (FFT), which holds a significant performance advantage over other methods on certain highly structured tasks, such as "date understanding" and "dyck language". The results suggest that only FFT enables the model to adequately learn the underlying structure and patterns required for these more complex and specialized tasks.

6.2 Discussion

Ablation study. Our instance-level approach significantly outperforms the task-level BatchComp, which directly selects LoRA modules without pairing questions to instances. BatchComp's 0.7% improvement over random LoraHub selection pales in comparison to our approach's doubling of performance in the "disambiguation qa" task, likely due to our method's superior ability to highlight the importance of key instances for task success.

Retrieval method	average
BM25	25.6
DPR L2 Distance	34.3
DPR Cosine Similarity	**35.6**

Table 3: Result of different retrieval strategy

Retrieval strategy. Our approach is highly dependent on retrieval performance. If accurate retrieval is not achieved, properly aligning suitable instances with corresponding questions and matching them with the appropriate LoRA modules, the overall effectiveness will be reduced, as demonstrated in Table 3 like BM25(Robertson et al., 1995). The results obtained from the DPR's L2 distance (Ram and Gray, 2012) and Cosine Similarity(Mussmann and Ermon, 2016) confirm the efficacy of DPR in instance-level fusion.

Time consumption. We evaluated the training and inference times for four methods—few-shot, LoraHub, BatchComp, and our Instance-Level method on a single RTX 3090 GPU. All inference tasks used a batch size of 10. The average time per batch was 2.1s for few-shot, 1.4s for LoraHub, 1.4s for BatchComp, and 3.7s for our method, respectively. Our method is 2.3 seconds slower than LoraHub because it requires training different parameters for each instance, which approximately necessitates N times the training resources for N instances. However, since inference utilizes the same resources, the time consumption is less than N times. Compared to the few-shot method, our method is only 1.6 seconds slower, as it, like LoraHub, does not require additional tokens for in-context learning.

Case study. Upon closer examination of the results for each task, we observed that tasks requiring reasoning, such as those performed with flan-t5-large, showed varying contributions to performance improvements based on the learning of task patterns. For instance, the Boolean expressions task, which has a standard "True or False" answer format, saw its accuracy increase to approximately 50% after training, compared to a significantly lower pre-training accuracy. Conversely, tasks involving date understanding, which could potentially mislead the model, exhibited shifts in outcomes for both zero-shot and ICL scenarios. Specifically, while the standard answer required selecting a response from given options, the inclusion of the MM/DD/YYYY format in the prompt led many responses to incorrectly provide dates like "08/03/1996" instead of choosing the correct option. However, methods that learned from individual instances were able to effectively avoid this type of misguidance.

7 Conclusion

Our work introduces two key enhancements to the LoraHub framework. The first is a method that indexes models trained on datasets based on their semantic centroids, improving LoraHub's precision at the task level. The second is instance-level adaptation, which leverages the unique characteristics of individual instances to raise the performance ceiling of the LoraHub approach. These complementary strategies work together to enhance the model's cross-task generalization capabilities.

8 Acknowledgements

This work was supported by the Strategic Priority Research Program of Chinese Academy of Sciences (No.XDA27020203) and the National Natural Science Foundation of China (No.62376270). This work was supported by the Youth Innovation Promotion Association CAS.

9 Limitation

Increased Computational Cost. Our method incurs a higher computational cost than LoraHub, mainly because we train weights for each individual instance during the LoRA group weights training phase. This means that our approach will require computational resources proportional to the number of instances, multiplied by the cost of LoraHub's training.

Application Scenario Limitation. Our method is not universally cost-effective. In scenarios where a task involves a limited number of questions, employing our method may not be the most economical choice. For tasks without any instances, zero-shot learning would be a more appropriate and efficient approach.

Additional Preliminary Preparations Required. When utilizing LoRA for composition, our method not only requires identifying the appropriate LoRA modules within the library but also necessitates access to the data used during the training of those LoRA modules. Consequently, our approach incurs greater initial preparation costs compared to methods that do not rely on such specific training data.

Requirement For Higher-Quality Instances. Instance-level methods, such as ours, are more sensitive to the quality of the instances used. Lower-quality instances, including those that are flawed or not closely related to the task, can potentially lead to misleading answers for associated questions. This underscores the importance of careful instance selection and curation to ensure the method's effectiveness.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

AI@Meta. 2024. Llama 3 model card.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *arXiv preprint*.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

Shangbin Feng, Weijia Shi, Yuyang Bai, Vidhisha Balachandran, Tianxing He, and Yulia Tsvetkov. 2023. Knowledge card: Filling llms' knowledge gaps with plug-in specialized language models. In *The Twelfth International Conference on Learning Representations*.

flax sentence embeddings. 2021. all_datasets_v4_minilm-l6 model card.

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. *Preprint*, arXiv:2106.09685.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2024. Lorahub: Efficient cross-task generalization via dynamic lora composition. *Preprint*, arXiv:2307.13269.

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

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020b. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

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. *arXiv preprint arXiv:2310.18339*.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, pages 22631–22648. PMLR.

Stephen Mussmann and Stefano Ermon. 2016. Learning and inference via maximum inner product search. In *International Conference on Machine Learning*, pages 2587–2596. PMLR.

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*.

Oded Ovadia, Menachem Brief, Moshik Mishaeli, and Oren Elisha. 2024. Fine-tuning or retrieval? comparing knowledge injection in llms. *Preprint*, arXiv:2312.05934.

Bohao Peng, Zhuotao Tian, Shu Liu, Mingchang Yang, and Jiaya Jia. 2024. Scalable language model with generalized continual learning. *arXiv preprint arXiv:2404.07470*.

Parikshit Ram and Alexander G Gray. 2012. Maximum inner-product search using cone trees. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 931–939.

Stephen Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. 1995. Okapi at trec-3. In *Overview of the Third Text REtrieval Conference (TREC-3)*, pages 109–126. Gaithersburg, MD: NIST.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671, Seattle, United States. Association for Computational Linguistics.

Sina Semnani, Violet Yao, Heidi Zhang, and Monica Lam. 2023. WikiChat: Stopping the hallucination of large language model chatbots by few-shot grounding on Wikipedia. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2387–2413, Singapore. Association for Computational Linguistics.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. 2022. Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning*, pages 20841–20855. PMLR.

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

Jihoon Tack, Jaehyung Kim, Eric Mitchell, Jinwoo Shin, Yee Whye Teh, and Jonathan Richard Schwarz. 2024. Online adaptation of language models with a memory of amortized contexts. *Preprint*, arXiv:2403.04317.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

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*.

Liang Wang, Nan Yang, and Furu Wei. 2024b. Learning to retrieve in-context examples for large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1752–1767, St. Julian's, Malta. Association for Computational Linguistics.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Preprint*, arXiv:2002.10957.

Xin Xu, Yue Liu, Panupong Pasupat, Mehran Kazemi, et al. 2024. In-context learning with retrieved demonstrations for language models: A survey. *arXiv preprint arXiv:2401.11624*.

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*.

A Result Detail And Stability Testing

This section presents our result details. Additionally, we conducted multiple random experiments, using random instances and random LoRA modules for LoraHub in each run to evaluate the stability of our method. We also tested the results with configurations other than the 5-shot, and the outcomes are as follows.

Method	zero-shot	ICL	FFT	LoraHub$_{avg}$	BatchComp	Ours
boolean expressions	35.9	25.7	53.5	48.2	46.1	49.8
causal judgement	58.8	58.2	58.8	58.8	57.7	59.9
date understanding	0.81	0.0	73.5	32.0	34.7	31.8
disambiguation qa	0.0	65.7	69.4	24.4	22.0	46.9
dyck languages	0.0	0.0	8.6	1.6	0.0	0.0
formal fallacies	55.1	52.7	52.2	53.1	52.2	53.5
geometric shapes	0.81	13.5	18.4	14.5	17.6	18.8
hyperbaton	26.5	0.41	48.2	68.6	69.8	71.8
logical deduction 5 objects	33.1	41.2	43.3	42.6	42.0	43.4
logical deduction 7 objects	33.5	38.0	47.4	44.4	41.2	40.8
logical deduction 3 objects	16.3	51.0	55.5	45.9	51.0	51.0
movie recommendation	49.8	42.4	64.5	53.1	52.7	50.2
multistep arithmetic two	0.0	0.0	0.0	0.5	0.0	0.4
navigate	56.3	59.6	57.1	53.5	58.8	56.3
object counting	26.5	26.9	34.7	27.9	28.6	31.4
penguins in a table	16.3	28.4	32.6	37.1	40.4	36.9
reasoning about colored objects	20.0	37.1	37.1	37.4	42.0	38.0
ruin names	22.0	26.1	57.1	21.9	22.4	22.0
salient translation error detection	29.0	42.0	20.0	31.6	30.2	31.0
snarks	48.6	43.9	48.0	52.2	58.4	58.4
sports understanding	4.1	53.5	45.3	50.1	50.2	46.5
temporal sequences	22.4	25.7	33.4	24.5	25.3	24.9
tracking shuffled objects 5 objects	11.0	10.6	16.7	11.0	11.0	11.0
tracking shuffled objects 7 objects	8.6	8.2	13.9	8.6	8.6	8.6
tracking shuffled objects 3 objects	31.0	31.8	34.3	31.0	32.2	32.2
web of lies	52.6	51.8	48.2	43.4	40.4	44.1
word sorting	0.81	0.0	3.7	0.95	0.81	0.81
average	24.4	30.9	39.8	34.0	34.7	35.6

Table 4: The results for the 27 tasks of BBH simulations have been obtained.

Method	zero-shot	ICL	FFT	LoraHub	BatchComp	Ours
boolean expressions	37.1	33.1	52.5	49.6	42.9	48.0
causal judgement	58.6	57.1	55.1	58.0	57.7	58.2
date understanding	0.8	2.3	55.9	28.8	25.2	23.9
disambiguation qa	0.0	65.2	57.7	13.1	23.5	33.7
dyck languages	0.0	0.0	11.4	1.1	0.7	0.1
formal fallacies	55.2	53.7	52.7	54.7	54.3	54.8
geometric shapes	0.8	9.8	25.0	14.4	16.9	19.5
hyperbaton	25.9	0.1	58.0	69.0	69.7	67.3
logical deduction 5 objects	33.1	41.1	43.1	42.9	43.1	42.2
logical deduction 7 objects	33.5	38.0	46.0	43.3	40.7	41.9
logical deduction 3 objects	16.5	50.7	54.1	50.4	50.7	53.7
movie recommendation	49.5	47.5	69.7	53.1	54.7	54.3
multistep arithmetic two	0.0	0.0	0.0	0.3	0.4	0.3
navigate	56.3	59.5	52.0	52.3	52.7	52.4
object counting	26.9	26.8	33.3	30.9	29.0	30.3
penguins in a table	16.3	24.3	37.6	38.1	39.7	38.1
reasoning about colored objects	20.0	37.7	35.6	39.0	40.1	39.9
ruin names	22.6	22.7	52.9	22.6	22.7	22.6
salient translation error detection	28.8	41.1	21.9	30.7	29.7	29.3
snarks	48.2	42.2	47.0	51.8	51.4	52.6
sports understanding	4.1	53.7	47.3	51.3	51.6	51.6
temporal sequences	21.6	25.9	36.2	24.5	23.0	23.1
tracking shuffled objects 5 objects	11.3	12.0	17.7	11.5	11.6	11.3
tracking shuffled objects 7 objects	8.4	7.9	14.1	8.4	8.4	8.4
tracking shuffled objects 3 objects	30.9	32.5	31.2	31.4	32.1	31.8
web of lies	52.7	51.4	49.4	51.0	24.2	48.0
word sorting	0.8	0.0	3.8	0.6	0.7	0.8
average	24.4	31.0	39.3	34.2	33.2	34.8

Table 5: The results for multiple random experiments.

Method	zero-shot	ICL	FFT	LoraHub	BatchComp	Ours
boolean expressions	0.88	5.84	3.89	3.02	3.28	1.50
causal judgement	0.26	0.90	2.92	1.51	0.45	1.35
date understanding	0.00	2.71	15.83	7.73	7.24	7.34
disambiguation qa	0.00	0.77	14.57	13.10	4.95	8.76
dyck languages	0.00	0.00	2.73	1.13	0.51	0.19
formal fallacies	0.19	1.02	0.33	1.28	1.45	1.02
geometric shapes	0.00	3.00	7.50	5.70	0.51	3.08
hyperbaton	0.51	0.19	8.00	4.79	0.19	3.05
logical deduction 5 objects	0.00	1.17	0.84	0.51	1.26	1.95
logical deduction 7 objects	0.00	0.33	1.92	3.79	5.68	1.35
logical deduction 3 objects	0.19	0.38	1.17	6.12	3.34	2.17
movie recommendation	0.19	4.00	3.72	3.12	3.79	2.65
multistep arithmetic two	0.00	0.00	0.00	0.17	0.33	0.19
navigate	0.33	1.50	7.60	6.39	6.74	5.89
object counting	0.33	1.17	1.02	1.87	1.53	0.38
penguins in a table	0.58	2.86	4.05	3.37	4.37	1.86
reasoning about colored objects	0.00	2.04	2.41	3.92	1.50	0.51
ruin names	0.51	2.67	2.99	0.48	0.38	0.51
salient translation error detection	0.19	0.69	7.78	2.80	1.71	1.71
snarks	0.27	1.25	1.79	5.02	4.90	4.03
sports understanding	0.00	0.38	1.67	2.30	1.02	2.55
temporal sequences	0.67	1.17	4.79	7.86	4.48	2.27
tracking shuffled objects 5 objects	0.19	1.02	0.84	0.45	0.51	0.19
tracking shuffled objects 7 objects	0.19	0.38	0.69	0.19	0.19	0.19
tracking shuffled objects 3 objects	0.19	0.51	2.50	0.87	0.84	0.58
web of lies	0.33	0.33	1.73	1.99	11.45	0.38
word sorting	0.00	0.00	0.19	0.34	0.19	0.00
average	0.02	0.06	0.65	0.90	1.05	0.72

Table 6: The standard deviation of the result from multiple random experiments.

#instance	avg score	standard deviation
1	30.6	0.35
3	34.2	0.38
5	34.8	0.72
10	35.6	0.81

Table 7: Average scores and standard deviations for different number of instances.