LoraHub: Efficient Cross-Task Generalization via Dynamic LoRA Composition

Chengsong Huang[†§*], **Qian Liu**[†*], **Bill Yuchen Lin**[◊*], **Tianyu Pang**[†], **Chao Du**[†], **Min Lin**[†]
[†]Sea AI Lab, Singapore
[§]Washington University in St. Louis, MO, USA
[◊]Allen Institute for AI, Seattle, WA, USA

Abstract

Low-rank adaptations (LoRA) are often employed to fine-tune large language models (LLMs) for new tasks. This paper investigates LoRA composability for cross-task generalization and introduces LoraHub, a strategic framework devised for the purposive assembly of LoRA modules trained on diverse given tasks, with the objective of achieving adaptable performance on unseen tasks. With just a few examples from a novel task, LoraHub enables the fluid combination of multiple LoRA modules, eradicating the need for human expertise and assumption. Notably, the composition requires neither additional model parameters nor gradients. Our empirical results, derived from the Big-Bench Hard (BBH) benchmark, suggest that LoraHub can effectively mimic the performance of in-context learning in few-shot scenarios, excluding the necessity of in-context examples alongside each inference input. A significant contribution of our research is the fostering of a platform for LoRA, where users can share their trained LoRA modules, thereby facilitating their application to new tasks. Code is available at `github.com/sail-sg/lorahub`.

1 Introduction

Significant progress in natural language processing has been largely fueled by large-scale pretrained language models (LLMs) such as OpenAI GPT (Brown et al., 2020), Flan-T5 (Chung et al., 2022), and LLaMA (Touvron et al., 2023). These models demonstrate top-tier performance across multiple NLP tasks. However, their enormous parameter size presents issues regarding computational efficiency and memory usage during fine-tuning. To mitigate these challenges, Low-Rank Adaptation (LoRA) (Hu et al., 2022) has emerged as a parameter-efficient fine-tuning technique (Lester et al., 2021; He et al., 2022; An et al., 2022). By reducing memory demands and computational costs, it speeds up LLM training. LoRA achieves this by freezing the base model parameters (that is, an LLM) and training a lightweight, ancillary module, which regularly delivers high performance on target tasks.

In this paper, we tap into the potential of LoRA modularity for broad task generalization, going beyond single-task training to meticulously compose LoRA modules for malleable performance on unknown tasks. Crucially, our method enables an automatic assembling of LoRA modules, eliminating dependency on manual design or human expertise. With just a handful of examples from unencountered tasks (e.g., 5), our approach can autonomously orchestrate compatible LoRA modules without human intrusion. As our approach leverages several available LoRA modules, we refer to it as LoraHub and denote our learning method as **LoraHub learning**.

To validate the efficiency of our proposed methods, we test our approaches using the widely recognized BBH benchmark with Flan-T5 (Chung et al., 2022) serving as the base LLM. The results underline the effectiveness of the LoRA module composition for unfamiliar tasks through a few-shot LoraHub

*The first three authors contributed equally to this work. Correspondence to Qian Liu at `liuqian@sea.com`.

R0-FoMo: Workshop on Robustness of Few-shot and Zero-shot Learning in Foundation Models at NeurIPS 2023.

learning process. Remarkably, our methodology achieves a score that closely matches the performance of few-shot, *in-context* learning. Additionally, our method substantially reduces the inference cost compared to in-context learning, eliminating the requirement of examples as inputs for the LLM. Our learning procedure is also notable for its computational efficiency, using a *gradient-free* approach to obtain the coefficients of LoRA modules and requiring only a handful of inference steps for unseen tasks. For instance, when applied to the BBH, our methodology can deliver superior performance in less than a minute using a single A100.

Importantly, LoraHub learning can feasibly be accomplished with a CPU-only machine, given that it merely requires proficiency to process LLM inference. With its versatility and robust performance, our work lays the foundation for the genesis of a platform, where users could effortlessly share, access, and apply trained LoRA modules to new tasks in this arena. Through this platform, we foresee the creation of a repository of versatile LoRA modules, fostering collaborative AI development. This allows the community to enhance the LLM's capabilities collectively through dynamic LoRA composition. Sharing and reusing modules optimizes resource utilization across diverse tasks.

2 Problem Statement

Large Language Models We assume that a large language model M_θ is based on Transformer architecture (Vaswani et al., 2017) and has been pre-trained on a large-scale natural language corpus. The model architecture can be either encoder-decoder (Raffel et al., 2020) or decoder-only (Brown et al., 2020). Also, M_θ could also have been fine-tuned with a large set of instruction-following datasets such as Flan Colleciton (Longpre et al., 2023) and PromptSource (Bach et al., 2022).

Cross-Task Generalization Assume we have N distinct *upstream tasks*, referred to as $\mathbb{T} = \{\mathcal{T}_1, ..., \mathcal{T}_N\}$. In practical situations, it is typical for users to desire an LLM to execute novel tasks that it has not encountered before— an ability widely known as *cross-task generalization*. Generally, cross-task generalization falls into two categories: zero-shot learning (Mishra et al., 2022; Sanh et al., 2022; Chung et al., 2022; OpenAI, 2022; Lin et al., 2022), which necessitates no labeled examples of the new task, and few-shot learning (Ye et al., 2021; Min et al., 2022) which demands a handful of labeled examples. Our paper principally concentrates on the latter, wherein for an unseen target task $\mathcal{T}' \notin \mathbb{T}$, users can only furnish a limited set of labeled examples, Q. Our aim is to modify the model M_θ to accustom itself to task $\mathcal{T}'$ using only Q. An intuitive method would be to directly fine-tune the weights of M_θ based on Q, yielding an updated model M_ϕ with enhanced performance on $\mathcal{T}'$. However, this approach is inefficient, time-consuming, and unstable when Q is small.

LoRA Tuning LoRA (Hu et al., 2022), a parameter-efficient fine-tuning method, facilitates the adaptation of LLMs using lightweight modules, eliminating the need for fine-tuning the entire weights. LoRA tuning involves keeping the original model weights frozen while introducing trainable low-rank decomposition matrices as adapter modules into each layer of the model. Compared to the base LLM, this module possesses significantly fewer trainable parameters, paving the way for rapid adaptation using minimal examples. As such, LoRA tuning presents a resource-efficient technique to quickly adapt LLMs for new tasks with restricted training data. However, traditional LoRA methods primarily concentrate on training and testing within the same tasks (Gema et al., 2023), rather than venturing into few-shot cross-task generalization.

3 Methodology

As depicted in Figure 1, we initially train LoRA modules on a variety of upstream tasks. Specifically, for N distinct upstream tasks, we separately train N LoRA modules, each represented as m_i for task $\mathcal{T}_i \in \mathbb{T}$. Subsequently, for a novel task $\mathcal{T}' \notin \mathbb{T}$, such as Boolean Expressions represented in Figure 1, its examples Q are utilized to steer the LoraHub learning process.

The LoraHub learning encapsulates two main phases: the COMPOSE phase and the ADAPT phase. In the COMPOSE phase, all available LoRA modules are synthesized into a single module $\hat{m}$, using $\{w_1, w_2, \ldots, w_N\}$ coefficients, represented as $\hat{m} = \sum_{i=1}^{N} w_i \times m_i$. Here, w_i is a scalar weight that can assume positive or negative values. During the ADAPT phase, the assembled LoRA module $\hat{m}$ is amalgamated with the LLM M_θ, and its performance on few-shot examples from the new task $\mathcal{T}'$ is assessed. A gradient-free algorithm is subsequently deployed to update w, enhancing $\hat{m}$'s performance (e.g., loss) on the few-shot examples Q.

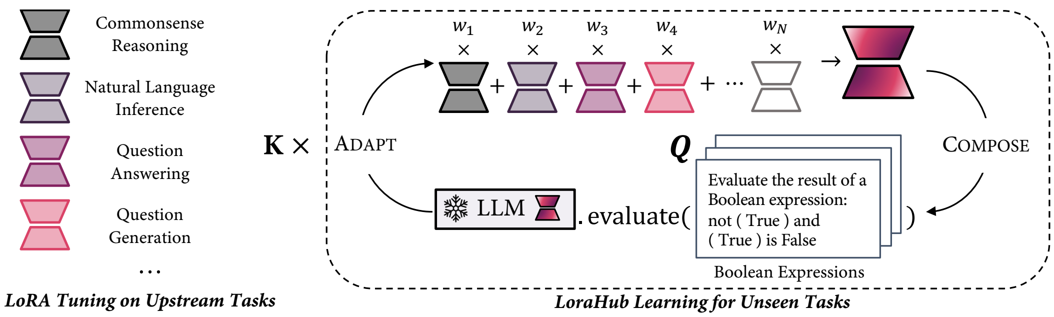

Figure 1: Our method encompasses two stages: the COMPOSE stage and the ADAPT stage. During the COMPOSE stage, existing LoRA modules are integrated into one unified module, employing a set of weights, denoted as w, as coefficients. In the ADAPT stage, the amalgamated LoRA module is evaluated on a few examples from the unseen task. Subsequently, a gradient-free algorithm is applied to refine w. After executing K iterations, a highly adapted LoRA module is produced, which can be incorporated with the LLM to perform the intended task.

Finally, after iterating through K steps, the optimum performing LoRA module is applied to the LLM M_θ, yielding the final LLM $M_\phi = \mathrm{LoRA}(M_\theta, \hat{m})$. This serves as an effectively adjusted model for the unseen task $\mathcal{T}'$, which will then be deployed and not updated anymore.

3.1 LoRA Tuning on Upstream Tasks

LoRA (Hu et al., 2022) effectively minimizes the number of trainable parameters through the process of decomposing the attention weight matrix update of the LLM, denoted as $W_0 \in R^{d\times k}$, into low-rank matrices. In more specific terms, LoRA exhibits the updated weight matrix in the form $W_0 + \delta W = W_0 + AB$, where $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times k}$ are novel low-rank matrices with rank r, a dimension significantly smaller than those of d and k. In this context, the product AB defines the LoRA module m, as previously elaborated. By leveraging this low-rank decomposition, LoRA imposes limitations that substantially reduce the number of trainable parameters necessary for adjusting the LLM weights.

3.2 COMPOSE: Element-wise Composition of LoRA Modules

In the COMPOSE stage, we implement an element-wise method for composing LoRA modules. This process integrates the corresponding parameters of the LoRA modules, necessitating that the modules being combined maintain an identical rank r in order to align the structures effectively. Given that $m_i = A_i B_i$, the combined LoRA module, denoted by $\hat{m}$, can be represented as:

$$\hat{m} = (w_1 A_1 + w_2 A_2 + \cdots + w_N A_N)(w_1 B_1 + w_2 B_2 + \cdots + w_N B_N). \tag{1}$$

3.3 ADAPT: Weight Optimization via Gradient-free Methods

During the ADAPT stage, our task is to fine-tune the coefficients w in order to boost the model's performace on a limited set of examples from a previously unseen task. One might think of using gradient descent for optimizing w, following standard backpropagation methods. However, this approach demands constructing a hypernet for all LoRA modules, reminiscent of differentiable architecture search methodologies (Zhang et al., 2019). This requirement for substantial GPU memory and time poses a challenge. Given that w consists of a relatively small number of parameters, we opted for gradient-free methods for optimization instead of gradient descent.

We utilize a black-box optimization technique following previous works (Sun et al., 2022) to find the optimal weights. The optimization process is steered by the cross-entropy loss, setting the goal to locate the best weight set $\{w_1, w_2, \ldots, w_N\}$ that reduces the loss L on the validation set Q. Furthermore, we incorporate L1 regularization to penalize the sum of the absolute values of all the ws, helping to prevent obtaining extreme values. Consequently, the final objective of LoraHub is to minimize $L + \alpha \cdot \sum_{i=1}^{N} |w_i|$, where α serves as a hyperparameter.

In most of the forthcoming experimental setups, we primarily employ the Covariance Matrix Adaptive Evolution Strategies (CMA-ES) (Hansen & Ostermeier, 1996). CMA-ES, as a stochastic, derivativefree, population-based optimization algorithm, offers versatility in addressing a broad spectrum of optimization challenges. It dynamically adjusts a search distribution, which is defined by a covariance matrix. During each iteration, CMA-ES systematically updates both the mean and covariance of this distribution to optimize the target function. In our specific application, we employ this algorithm to mold the search space for the variable w. Ultimately, we use it to identify the optimal weights by evaluating their performance on a few-shot examples from an unseen task.

4 Evaluation

4.1 Experimental Framework

Large Language Model We employ Flan-T5 (Chung et al., 2022) as our chosen Large Language Model (LLM). This series of models, characterized by similar structures and varying sizes, exhibits exemplary zero-shot and few-shot capabilities commensurate with the model size. Our investigation particularly targets the Flan-T5 large model.

Candidate LoRA Modules Our methodology requires a compendium of LoRA modules trained on preceding tasks. For parity with Flan, we adopt the tasks utilized to instruct Flan-T5, thereby incorporating nearly 200 distinct tasks and their corresponding instructions [2]. Following this, we created several LoRA modules as possible candidates [3]. For the pre-filtering process, during each experimental sequence, we randomly select 20 LoRA modules for potential considerations.

Dataset and Evaluation We employ 27 tasks from he Big-Bench Hard (BBH) benchmark, following the challenging stipulations for language models outlined by previous researchers. Throughout all tasks, we employ Exact Match (EM) as our evaluation metric.

Baseline Setup To enhance the demonstration of our method's performance, we expanded our comparisons beyond the zero-shot and in-context learning settings. We specifically chose three representative gradient-based methods for comparison: full fine-tuning (FFT), LoRA tuning (LoRA), and IA3 fine-tuning (IA3) (Liu et al., 2022). For all gradient-based methods, for a fair comparsion, we train for 40 epochs on the same three runs of 5 examples employed in our methods. In the case of FFT, a learning rate of 3e-5 is employed, whereas for IA3 and LoRA, we adopt a learning rate of 2e-4. We report the performance of each method on the test set at the end of training (averaged over three runs) without any model selection to avoid potential selection bias.

4.2 Implementation Details

We implemented LoRA tuning using the Huggingface PEFT library (Mangrulkar et al., 2022), keeping the default LoRA tuning hyperparameter at $r = 16$. The gradient-free method was implemented using the open-source Nevergrad optimization library (Rapin & Teytaud, 2018), imposing a constraint that the absolute value of LoRA weights should not exceed 1.5. At the outset, all LoRA modules were set at zero weights. In our standard settings, we permitted a maximum of 40 attempts to calculate the loss on the samples, denoted by the maximum step size K. Five examples were used during optimization, the same number as used in the few-shot in-context learning scenario. And the hyperparameter α is set as 0.05.

4.3 Main results

As shown in Table 1, our experimental results demonstarte the superior efficacy of our method in comparison to zero-shot learning while closely resembling the performance of in-context learning (ICL) in few-shot scenarios. This observation is derived from an average performance of three runs, each leveraging different few-shot examples. Importantly, our model utilizes an equivalent number of tokens as the zero-shot method, notably fewer than the count used by ICL. Although occasional performance fluctuations, our method consistently outperforms zero-shot learning in most tasks. In the era of LLMs, the input length is directly proportional to the inference cost, and thus LoraHub's ability to economize on input tokens while approaching the peak performance grows increasingly

[2] We accessed these publicly available tasks via huggingface.co/datasets/conceptofmind/FLAN_2022.

[3] These LoRA modules can be accessed at huggingface.co/models?search=lorahub.

Table 1: Experimental results of zero-shot learning (Zero), few-shot in-context learning (ICL), IA3 fine-tuning (IA3), LoRA tuning (LoRA), full fine-tuning (FFT) and our proposed few-shot LoraHub learning (LoraHub) on the BBH benchmark with FLAN-T5-large as the base LLM. We denote algorithmic tasks with the superscript § following previous work (Wu et al., 2023). Note that we employ three runs, each leveraging different 5-shot examples per task, as demonstrations for all few-shot methods. The average performance of all methods is reported below, and the best performance of each few-shot method can be found in the Appendix A.

Task	Zero	ICL_{avg}	IA3_{avg}	LoRA_{avg}	FFT_{avg}	$\text{LoraHub}_{\text{avg}}$
Boolean Expressions	54.0	59.6	56.2	56.0	62.2	55.5
Causal Judgement	57.5	59.4	60.2	55.6	57.5	54.3
Date Understanding	15.3	20.4	20.0	35.8	59.3	32.9
Disambiguation	0.0	69.1	0.0	68.0	68.2	45.2
Dyck Languages	1.3	0.9	4.2	22.2	19.5	1.0
Formal Fallacies	51.3	55.3	51.5	53.6	54.0	52.8
Geometric Shapes	6.7	19.6	14.7	24	31.1	7.4
Hyperbaton	6.7	71.8	49.3	55.3	77.3	62.8
Logical Deduction§ (five objects)	21.3	39.1	32.7	40.0	42.2	36.1
Logical Deduction§ (seven objects)	12.7	40.7	33.8	37.3	44.9	36.8
Logical Deduction§ (three objects)	0.0	51.6	8.5	53.6	52.9	45.7
Movie Recommendation	62.7	55.8	61.8	51.5	66.0	55.3
Multistep Arithmetic	0.7	0.7	0.7	0.2	0.0	0.4
Navigate	47.3	45.3	46.2	48.0	48.0	47.1
Object Counting	34.7	32.4	35.1	38.7	35.6	33.7
Penguins in a Table	43.5	41.3	45.0	36.2	31.9	35.9
Reasoning about Colored Objects	32.0	40.2	40.7	39.6	37.6	40.0
Ruin Names	23.3	19.3	24.4	37.8	61.3	24.4
Salient Translation Error Detection	37.3	47.3	37.1	16.0	16.2	36.0
Snarks	50.0	54.2	53.9	55.6	66.7	56.9
Sports Understanding	56.0	54.7	55.1	56.5	54.0	56.7
Temporal Sequences	16.7	25.1	18.2	25.1	37.8	18.2
Tracking Shuffled Objects§ (five objects)	12.0	12.0	12.0	13.8	16.9	12.3
Tracking Shuffled Objects§ (seven objects)	6.7	6.7	6.7	10.0	9.8	7.7
Tracking Shuffled Objects§ (three objects)	24.7	31.1	30.7	30.9	32.0	29.2
Web of Lies	54.0	53.8	54.2	52.7	48.2	50.1
Word Sorting	1.3	0.5	1.3	4.9	4.9	1.1
Avg Performance Per Task	27.0	37.3	31.6	37.7	42.1	34.7
Avg Tokens Per Example	111.6	597.8	111.6	111.6	111.6	111.6
Gradient-based Training	No	No	Yes	Yes	Yes	No

significant. Moreover, as shown in Appendix Table 2, the upper bound performance of our method across these runs can surpass ICL on 18 tasks, demonstrating its potential for future development.

Even when compared to certain gradient-based optimization methods, our approach consistently demonstrates competitive performance. For example, as depicted in Table 1, our method exhibits a notable improvement of 3.1% on average in contrast to the promising IA3 method. Nevertheless, we acknowledge that our approach still falls behind LoRA tuning and full fine-tuning, especially in tasks that exhibit significant deviation from the upstream task. Taking Dyck Languages as an example, both LoraHub and ICL achieve only an average performance of nearly 1.0% on these tasks, while LoRA and FFT methods showcase impressive results with only 5 examples.

4.4 Discussion

LoraHub addresses the challenge of reducing inference costs by eliminating the need for processing additional tokens, resulting in a noticeable reduction in overall inference expenses. However, it introduces an inherent cost during the ADAPT stage, necessitating extra inference steps, such as the 40 steps employed in our experiments. This introduces a trade-off between choosing the ICL approach and LoraHub, with the decision typically hinging on the nature of the situation.

For one-time ad-hoc tasks, the ICL approach should be more pragmatic due to LoraHub's additional inference step costs. In such scenarios, where immediate, single-use solutions are preferred, the simplicity and efficiency of ICL might outweigh the benefits of potential savings offered by LoraHub. Conversely, for recurring or similar tasks, LoraHub emerges as a compelling option. Despite the added inference step cost, LoraHub's ability to efficiently handle repetitive tasks, often occurring thousands of times, while concurrently reducing overall expenses, positions it as a viable option in such kind of situations.

In summary, our intention is not to replace ICL, but to present LoraHub as a complementary strategy with performance-efficiency trade-offs. Thus, we encourage a careful consideration of specific use cases and requirements when choosing between ICL and LoraHub, recognizing that the optimal solution may vary based on the nature and frequency of the tasks at hand.

5 Related Work

Model Merging Our method substantially draws on the concept of LoRA module composition, and thus, aligns with the significant thread of research in model merging. This research focus is broadly categorized based on the ultimate objectives of model merging.

For merging entire models, models are combined to approximate ensemble or multi-task learning benefits. Previous works like Matena & Raffel (2021) and Jin et al. (2023) assumed shared architectures. Matena & Raffel (2021) uses Gaussian posterior distributions from Fisher information, while Jin et al. (2023) merges models by minimizing prediction differences. We can also merge models with varying architectures. For example, Ainsworth et al. (2023) adjusts model weights before merging, while Stoica et al. (2023) identifies common features for merging models on different tasks. In contrast, our work focuses on cross-task generalization through model merging.

Module Merging The second category most closely aligns with our research, stemming from a shared motivation of module composition. Various scholars have made advances in this line of research: Kingetsu et al. (2021) decomposes and recomposes modules on the basis of their functionality; Ilharco et al. (2022) proposes modulating model behavior using task vectors; Wang et al. (2022) Lv et al. (2023) amalgamates parameter-efficient modules weighted according to task similarity; Zhang et al. (2023) crafts modules by employing specific arithmetic operations; Sun et al. (2023) improves few-shot performance of unseen tasks by multi-task pre-training of prompts; Chronopoulou et al. (2023) averages adapter weights intended for transfer; and Muqeeth et al. (2023) concentrates on amalgamating experts in mixture of experts models. However, these methods generally necessitate multi-task training or human prior on module selection for the downstream task.

Mixture of Experts The Mixture of Experts (MoE) is an ensemble method, often visualized as a collection of sub-modules, or 'experts', each specializing in processing different types of input data. Each expert in this system is controlled by a unique gating network, activated based on the distinct nature of the input data. This technique has proven instrumental in numerous domains, such as natural language processing and computer vision (Jacobs et al., 1991; Shazeer et al., 2017; Du et al., 2022; Zhang et al., 2022). Our methodology displays similarities to MoE, wherein upstream-trained LoRA modules can be aligned with MoE's expert design. A noteworthy distinguishing factor is that our approach mechanism does not require any specialized manipulation of LoRAs during training while facilitating dynamic LoRA module assembly at any scale, each pre-tuned to different tasks. Recent studies on the interrelation between MoE and instruction tuning have demonstrated that the simultaneous application of both approaches enhances the effectiveness of each individually (Shen et al., 2023).

6 Conclusion

In this work, we have introduced LoraHub, a strategic framework for composing LoRA modules trained on diverse tasks in order to achieve adaptable performance on new tasks. Our approach enables the fluid combination of multiple LoRA modules using just a few examples from a novel task, without requiring additional model parameters or human expertise. The empirical results on the BBH benchmark demonstrate that LoraHub can effectively match the performance of in-context learning in few-shot scenarios, removing the need for in-context examples during inference. Overall, our work shows the promise of strategic LoRA composability for rapidly adapting LLMs to diverse tasks. By fostering reuse and combination of LoRA modules, we can work towards more general and adaptable LLMs while minimizing training costs.

Acknowledgement

We would like to thank all the anonymous reviewers for their constructive feedback.

References

Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=CQsmMYmlP5T.

Shengnan An, Yifei Li, Zeqi Lin, Qian Liu, Bei Chen, Qiang Fu, Weizhu Chen, Nanning Zheng, and Jian-Guang Lou. Input-tuning: Adapting unfamiliar inputs to frozen pretrained models. *ArXiv preprint*, abs/2203.03131, 2022. URL https://arxiv.org/abs/2203.03131.

Stephen Bach, Victor Sanh, Zheng Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-david, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Fries, Maged Al-shaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Dragomir Radev, Mike Tian-jian Jiang, and Alexander Rush. PromptSource: An integrated development environment and repository for natural language prompts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 93–104, Dublin, Ireland, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-demo.9. URL https://aclanthology.org/2022.acl-demo.9.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Alexandra Chronopoulou, Matthew Peters, Alexander Fraser, and Jesse Dodge. AdapterSoup: Weight averaging to improve generalization of pretrained language models. In *Findings of the Association for Computational Linguistics: EACL 2023*, pp. 2054–2063, Dubrovnik, Croatia, 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.findings-eacl.153.

Hyung Won Chung, Le Hou, S. 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, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Wei Yu, Vincent Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed Huai hsin Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *ArXiv preprint*, abs/2210.11416, 2022. URL https://arxiv.org/abs/2210.11416.

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P. Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen S. Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V. Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. Glam: Efficient scaling of language models with mixture-of-experts.

In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 5547–5569. PMLR, 2022. URL https://proceedings.mlr.press/v162/du22c.html.

Aryo Pradipta Gema, Luke Daines, Pasquale Minervini, and Beatrice Alex. Parameter-efficient fine-tuning of llama for the clinical domain. *ArXiv preprint*, 2023.

Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 312–317, 1996.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=0RDcd5Axok.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *ArXiv preprint*, abs/2212.04089, 2022. URL https://arxiv.org/abs/2212.04089.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991. URL https://direct.mit.edu/neco/article-abstract/3/1/79/5560/Adaptive-Mixtures-of-Local-Experts?redirectedFrom=fulltext.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=FCnohuR6AnM.

Hiroaki Kingetsu, Kenichi Kobayashi, and Taiji Suzuki. Neural network module decomposition and recomposition. *ArXiv preprint*, abs/2112.13208, 2021. URL https://arxiv.org/abs/2112.13208.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.243. URL https://aclanthology.org/2021.emnlp-main.243.

Bill Yuchen Lin, Kangmin Tan, Chris Miller, Beiwen Tian, and Xiang Ren. Unsupervised cross-task generalization via retrieval augmentation. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/8a0d3ae989a382ce6e50312bc35bf7e1-Abstract-Conference.html.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *ArXiv*, abs/2205.05638, 2022. URL https://api.semanticscholar.org/CorpusID:248693283.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. The flan collection: Designing data and methods for effective instruction tuning, 2023.

Xingtai Lv, Ning Ding, Yujia Qin, Zhiyuan Liu, and Maosong Sun. Parameter-efficient weight ensembling facilitates task-level knowledge transfer. In *Annual Meeting of the Association for Computational Linguistics*, 2023.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, and Sayak Paul. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022.

Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging. *ArXiv preprint*, abs/2111.09832, 2021. URL https://arxiv.org/abs/2111.09832.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. MetaICL: Learning to learn in context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2791–2809, Seattle, United States, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.201. URL https://aclanthology.org/2022.naacl-main.201.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3470–3487, Dublin, Ireland, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.244. URL https://aclanthology.org/2022.acl-long.244.

Mohammed Muqeeth, Haokun Liu, and Colin Raffel. Soft merging of experts with adaptive routing. *ArXiv preprint*, abs/2306.03745, 2023. URL https://arxiv.org/abs/2306.03745.

OpenAI. ChatGPT. 2022. URL https://openai.com/blog/chatgpt.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

J. Rapin and O. Teytaud. Nevergrad - A gradient-free optimization platform. https://GitHub.com/FacebookResearch/Nevergrad, 2018.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=9Vrb9D0WI4.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Sheng Shen, Le Hou, Yanqi Zhou, Nan Du, Shayne Longpre, Jason Wei, Hyung Won Chung, Barret Zoph, William Fedus, Xinyun Chen, Tu Vu, Yuexin Wu, Wuyang Chen, Albert Webson, Yunxuan Li, Vincent Zhao, Hongkun Yu, Kurt Keutzer, Trevor Darrell, and Denny Zhou. Mixture-of-experts meets instruction tuning:a winning combination for large language models, 2023.

George Stoica, Daniel Bolya, Jakob Bjorner, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training. *arXiv*, 2023.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. Black-box tuning for language-model-as-a-service. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 20841–20855. PMLR, 2022. URL https://proceedings.mlr.press/v162/sun22e.html.

Tianxiang Sun, Zhengfu He, Qin Zhu, Xipeng Qiu, and Xuanjing Huang. Multitask pre-training of modular prompt for Chinese few-shot learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11156–11172, Toronto, Canada, July 2023. Association for Computational Linguistics. URL `https://aclanthology.org/2023.acl-long.625`.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *ArXiv preprint*, abs/2302.13971, 2023. URL `https://arxiv.org/abs/2302.13971`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html`.

Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. AdaMix: Mixture-of-adaptations for parameter-efficient model tuning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 5744–5760, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.388. URL `https://aclanthology.org/2022.emnlp-main.388`.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David S. Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance. *CoRR*, abs/2303.17564, 2023. doi: 10.48550/arXiv.2303.17564. URL `https://doi.org/10.48550/arXiv.2303.17564`.

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. CrossFit: A few-shot learning challenge for cross-task generalization in NLP. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7163–7189, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.572. URL `https://aclanthology.org/2021.emnlp-main.572`.

Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL `https://openreview.net/forum?id=rkgW0oA9FX`.

Fan Zhang, Duyu Tang, Yong Dai, Cong Zhou, Shuangzhi Wu, and Shuming Shi. Skillnet-nlu: A sparsely activated model for general-purpose natural language understanding, 2022.

Jinghan Zhang, Shiqi Chen, Junteng Liu, and Junxian He. Composing parameter-efficient modules with arithmetic operations. *ArXiv preprint*, abs/2306.14870, 2023. URL `https://arxiv.org/abs/2306.14870`.

A Result of Best Results

As shown in Table 2, compared to gradient-based parameter-efficient training methods like LoRA and IA3, our approach demonstrates superior performance in terms of best results over experimental runs. While it exhibits a noticeable lag behind the fully fine-tuning (FFT) method, which updates all parameters during training, this observation suggests that our proposed method has a promising upper limit. We anticipate that future research efforts can contribute to accelerating the optimization speed and further enhancing the efficacy of our approach.

Table 2: Experimental results of several few-shot methods, including in-context learning (ICL), IA3 fine-tuning (IA3), LoRA tuning (LoRA), full fine-tuning (FFT) and our LoraHub learning (LoraHub) on the BBH benchmark with FLAN-T5-large as the base LLM. We denote algorithmic tasks with the superscript § following previous work (Wu et al., 2023). Note that we use 5 examples per task as the demonstration for all methods. The best (*best*) performance is reported as the maximum value obtained across three runs.

Task	ICL_{best}	IA3_{best}	$\text{LoRA}_{\text{best}}$	FFT_{best}	$\text{LoraHub}_{\text{best}}$
Boolean Expressions	62.7	58.0	60.7	65.3	60.7
Causal Judgement	59.8	62.1	57.5	60.9	63.2
Date Understanding	21.3	20.7	40.7	67.3	45.3
Disambiguation	69.3	0.0	68.7	70.7	68.0
Dyck Languages	2.0	4.7	25.3	33.3	2.7
Formal Fallacies	59.3	52.0	56.7	56.0	59.3
Geometric Shapes	20.0	15.3	28.7	39.3	18.7
Hyperbaton	72.7	49.3	57.3	82.0	72.7
Logical Deduction§ (five objects)	39.3	32.7	41.3	43.3	40.0
Logical Deduction§ (seven objects)	42.0	34.0	42.7	46.0	46.0
Logical Deduction§ (three objects)	52.7	8.7	56.7	60.7	52.7
Movie Recommendation	56.7	62.0	64.5	70.7	62.0
Multistep Arithmetic	0.7	0.7	0.7	0.0	1.3
Navigate	46.7	47.3	50.7	50.0	51.3
Object Counting	34.7	35.3	42.0	38.0	36.7
Penguins in a Table	43.5	45.7	41.3	37.0	47.8
Reasoning about Colored Objects	41.3	41.3	40.7	38.7	44.7
Ruin Names	20.7	25.3	42.0	66.0	28.7
Salient Translation Error Detection	48.0	37.3	17.3	21.3	42.7
Snarks	55.1	56.4	59.0	69.2	61.5
Sports Understanding	56.7	55.3	58.7	58.7	62.7
Temporal Sequences	26.7	18.7	31.3	48.7	21.3
Tracking Shuffled Objects§ (five objects)	12.0	12.0	16.0	20.0	16.7
Tracking Shuffled Objects§ (seven objects)	6.7	6.7	12.0	10.0	15.3
Tracking Shuffled Objects§ (three objects)	31.3	30.7	32.0	36.0	31.3
Web of Lies	54.0	54.7	55.3	54.0	57.3
Word Sorting	0.7	1.3	5.3	6.0	1.3
Best Performance (Average)	38.4	32.1	40.9	46.2	41.2