LoRA Soups: Merging LoRAs for Practical Skill Composition Tasks

Akshara Prabhakar
Department of Computer Science
Princeton University

Yuanzhi Li
Microsoft Research

Karthik Narasimhan
Department of Computer Science
Princeton University

Sham Kakade, Eran Malach
Kempner Institute
Harvard University

Samy Jelassi
Center of Mathematical Sciences and Applications
Harvard University

Abstract

Low-Rank Adaptation (LoRA) is a popular technique for parameter-efficient fine-tuning of Large Language Models (LLMs). We study how different LoRA modules can be merged to achieve *skill composition*—testing the performance of the merged model on a target task that involves combining multiple skills, each skill coming from a single LoRA. This setup is favorable when it is difficult to obtain training data for the target task and when it can be decomposed into multiple skills. First, we identify practically occurring use-cases that can be studied under the realm of skill composition, e.g. solving hard math-word problems with code, creating a bot to answer questions on proprietary manuals or about domain-specialized corpora. Our main contribution is to show that concatenation of LoRAs (**CAT**), which optimally weights LoRAs that were individually trained on different skills, outperforms existing model- and data- merging techniques; for instance on math-word problems, CAT beats these methods by an average of 43% and 12% respectively. Thus, this paper advocates model merging as an efficient way to solve compositional tasks and underscores CAT as a simple, compute-friendly and effective procedure.[*] [†]

1 Introduction

Large Language Models (LLMs) have demonstrated impressive capabilities in conversational tasks and general-purpose applications, such as writing emails or answering common questions. However, these general purpose LLMs may have restricted performance on tasks where specific skills and knowledge is required. We primarily focus on skill composition tasks, that necessitate the integration of multiple skills.

Many industrial applications fit in this framework. Consider a company that manufactures ovens and is trying to design a chatbot to answer customer queries about its working and specifics. Directly using a frontier LLM (like `gpt-4o`) would fail since it lacks knowledge about the company's product. The ideal solution here would be to design an instruction dataset consisting of question-answer pairs about this product and fine-tuning an LLM on it. However, such a data collection and annotation procedure is expensive. Another possible solution is to fine-tune an LLM on a collection of product manuals and then impart it chat abilities by further fine-tuning on an instruction-tuning dataset like Alpaca (Taori et al., 2023). We refer to this method as DATA-MIX. Besides this approach being sequential, it suffers from catastrophic forgetting (Kirkpatrick et al., 2017). Whenever the company creates a new product, they need to redo fine-tuning on this data mixture or create a new question-answer dataset for the former method.

In this paper, we study *model merging* as an alternative approach. Given a model that is fine-tuned on the manuals and one that possesses question-answering capabilities, we optimally combine their weights to obtain a model that can answer product-specific questions. This approach is more efficient since we merge skill-specific fine-tuned models without any additional data collection or training from scratch. Among the multiple techniques to perform model merging, our framework specifically builds on LoRA[‡](Hu et al., 2021), a fine-tuning technique that consists of adding a low-rank update to a few layers in the model. In this context, model merging consists of combining LoRA weights from different models.

Given LoRAs trained on specialized domains (biology, math, code, reading comprehension, question-answering), is it possible to merge them to

[*]Code and data are available at https://github.com/aksh555/LoRA-Soups.

[†]Extended preprint version: https://arxiv.org/abs/2410.13025.

[‡]See Appendix A for a review of the LoRA method.

Proceedings of the 31st International Conference on Computational Linguistics: Industry Track, pages 644–655
January 19–24, 2025.

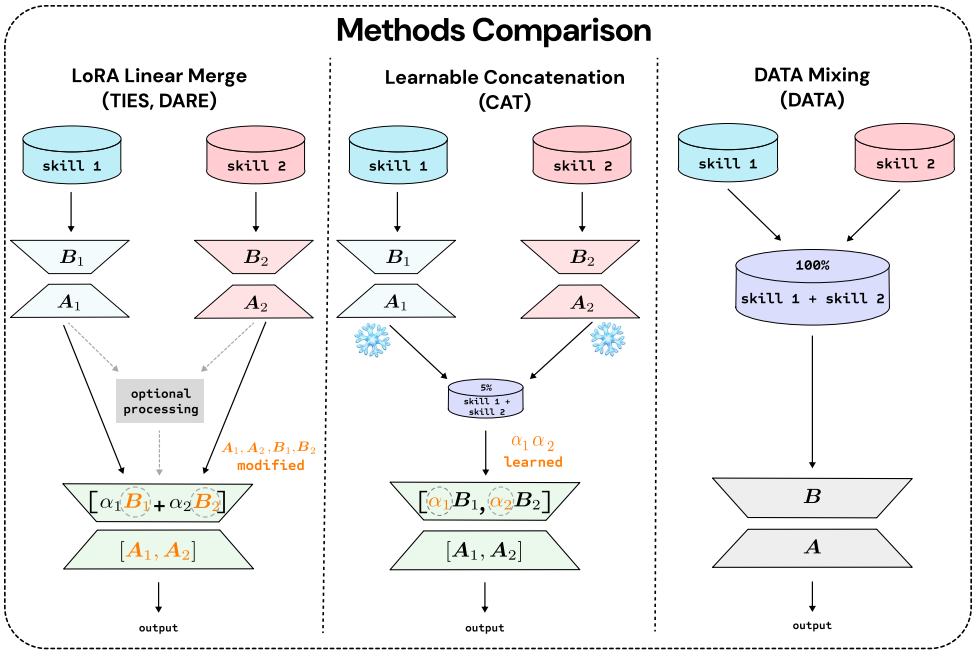


Figure 1: Comparison of Learnable Concatenation (**CAT**) with Linear and DATA-MIX methods.

effectively solve a new problem that requires a combination of these domains? We underline that most settings we consider are *out-of-domain* since the specialized LoRAs have been trained on datasets that are very different from the target task. To our knowledge, this is the first paper exhibiting model merging is superior to data mixing for binary skill composition problems.

Our key contributions are summarized:

– In section 3, we analyze practical applications in domains spanning code, science, robustness, and in-house use cases under the purview of binary *skill composition*.

– We introduce Learnable Concatenation (**CAT**), a LoRA merging technique that involves a simple weighted average of the encompassed skill LoRAs.

– In section 5, we perform a comprehensive evaluation of several baselines and demonstrate that **CAT** achieves better binary skill composition than both existing model merging methods and data mixing.

2 Related Work

Merging methods. The traditional approach to learning multiple skills/tasks simultaneously is joint training on a mixture of task datasets (Caruana, 1997). As data collection for specialized tasks and training large models from scratch get more expensive; coupled with the rapid expansion in the availability of well-trained open-source models – model merging has emerged as a convenient way of building powerful models from existing ones (Goddard et al., 2024; Leroo-AI, 2024). The richly studied simplest way of merging by averaging model weights (Utans, 1996; Smith and Gashler, 2017; Garipov et al., 2018; Izmailov et al., 2018) paved the way to linear weight averaging (Wortsman et al., 2022). Expanding on weight averaging, Task Arithmetic (Ilharco et al., 2022) involving the creation and combination of task vectors facilitated multi-task learning. While this weight interpolation was heavily used for merging image generation models, recent methods like TIES (Yadav et al., 2024) and DARE (Akiba et al., 2024) reset redundant parameters, resolve sign conflicts, and exclusively merge parameters that exhibit sign-consistency, and SLERP (White, 2017) by spherical linear interpolation build upon this for language models. In these methods, the coefficients governing the model merging are determined by trial-error. In contrast to this, CAT learns these coefficients layer-wise cheaply.

LoRA merging methods. Recently, the vision community witnessed the widespread application LoRAs (Buehler and Buehler, 2024a; Feng et al., 2024; Luo et al., 2024; Muqeeth et al., 2024; Wu et al., 2023; Zhong et al., 2024; Yang et al., 2024a) as an effective approach to multi-task learning and composing styles and subjects (Shah et al., 2023).

Many of these utilize Mixture of Experts (MoE) (Buehler and Buehler, 2024a; Feng et al., 2024; Luo et al., 2024; Wu et al., 2023) based architectures having input-dependent learnable routers. These models have been primarily used in the context of multitask learning. In this work, we study LoRA model merging for *skill composition* tasks.

LoRA merging for multitask learning and compositionality. Prior works (Gu et al., 2024; Shah et al., 2023; Yang et al., 2024b) have investigated LoRA merging in computer vision where each skill is a visual concept or style and the objective is image generation. On the other hand, natural language tasks are more challenging since identifying the skills needed for solving a task is not always clear. On natural language tasks, most prior works (Buehler and Buehler, 2024a; Feng et al., 2024; Luo et al., 2024; Muqeeth et al., 2024; Wu et al., 2023) merged LoRAs with the objective of multitask learning. In this setting, the individual LoRA modules are trained on (potentially) independent tasks and the merged model is tested on the original tasks. A successful model retains the skills of each individual LoRA. Differently, Huang et al. (2023) devise LoraHub, a strategy to merge LoRAs for cross-task learning. By finetuning LoRAs on FLAN (Longpre et al., 2023), they achieve performance equivalent to few-shot prompting on some Big-Bench Hard (BBH) tasks (Suzgun et al., 2022). Closer to our work, Akiba et al. (2024) merge specialized LMs in Japanese and in math (Cobbe et al., 2021) to solve word-math problems in Japanese (Shi et al., 2022). Though this can be viewed as a skill composition task, they focus on studying only one such task, and their contribution is an evolutionary algorithm for merging models.

3 Skill Composition

Most of the downstream tasks used to evaluate LLMs require mastering multiple skills to be solved. Skill here refers to specific capabilities that the LLM needs for customization to downstream use cases. These skills can be acquired from knowledge source like textbooks and manuals or from foundational datasets designed for arithmetic, coding, etc. For instance, achieving high score in the GSM8k benchmark (Cobbe et al., 2021) requires good commonsense reasoning and arithmetic skills. In this paper, we focus on downstream tasks where composition can be ensured and isolated, and we mainly focus on tasks that require two skills. Skill composition is challenging because, not only does the model need to "know" the different skills, but it also needs to understand the appropriate context for applying each skill. We present some skill composition examples of practical interest in Figure 2.

Hard math-word problems. Prior works noticed that when fine-tuning LLMs on GSM-8k (Cobbe et al., 2021), the resulting model performs poorly on GSM-Hard (Gao et al., 2023), that gathers similar problems but with more complex arithmetic operations. The program-aided approach (Gao et al., 2023) is one solution to address this issue: The model takes in a math word problem, mathematically reasons, and then outputs a corresponding Python function whose return value is the answer to the problem. For this, the model must excel in mathematical skills to first reason about the problem and also coding skills to translate its reasoning to code. Therefore, to improve the accuracy on GSM-Hard, we set the first skill to be math reasoning and the second skill to be coding.

QABot on proprietary manuals. While general-purpose chatbots are useful, an institution or corporation may desire to have a *specialized* question-answering bot/assistant that addresses domain-specific questions. Examples of this case may be a university that wants a bot that answers questions about quantum physics or a refrigerator retailer that wants a bot to answer questions about their device. The traditional approach may fall short as fine-tuning a chat model on a specialized document may cause loss in its conversational abilities. Another solution would be to obtain an instruction-tuning dataset that covers the specialized material but this is very expensive. On the other hand, model merging seems to be a convenient solution to address this problem: we train one LoRA on a general instruction-tuning dataset and another one on the specialized document.

Reading comprehension on technical documents. Medical reports and law contracts are very challenging to read for non-specialist users: they are usually very long and involve a myriad of technical jargon. Having a model that can read these documents and answer questions would be very useful. For this reason, we train one LoRA to acquire the reading comprehension skill and another LoRA on the domain-specific documents.

Robustness to variations in prompt format. Recently, a few works (Mizrahi et al., 2024; Sclar

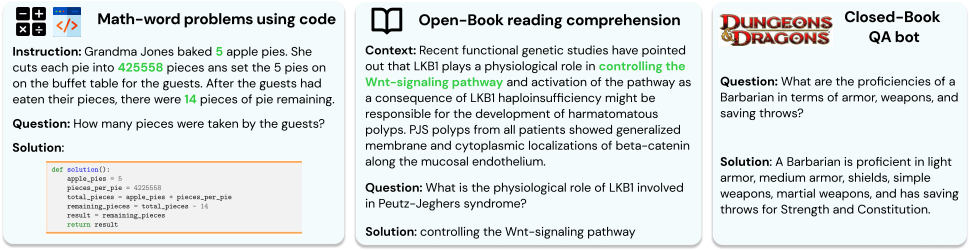


Figure 2: Examples of binary skill composition tasks.

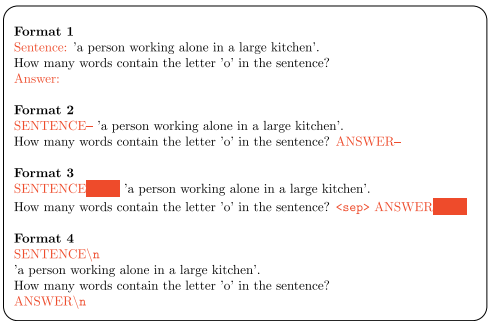


Figure 3: Different prompt formats – varying *descriptors*, *separators*, and *spaces* for the same task example.

et al., 2024) reported performance differences of up to 76 accuracy points due to subtle changes in prompt formatting when using Llama-2-13B (Touvron et al., 2023). This sensitivity remains when increasing model size, the number of few-shot examples, or performing instruction tuning. We investigate whether model or data mixing is a solution to this problem. Therefore, we set the first skill to be the dataset with format i and the second skill to be the dataset with format j and evaluate on a dataset with format k, where $i, j, k \in \{1, \ldots, 10\}$ and $i \neq j \neq k$. Figure 3 shows an example.

3.1 Merging methods

Figure 1 details the various merging methods in the literature and the proposed CAT method.

Data mixing (DATA-MIX). The naive approach for solving these tasks is to train on a mixture of datasets containing different skills. We fine-tune a *single* model with LoRA weights (A, B) where $A, B \in \mathbb{R}^{d \times kr}$ (rank is kr (and not r) in order to ensure a fair comparison with the other merging methods) on the concatenation of datasets and thus simultaneously teach the k skills to the model.

Model merging. Here, we train one model per skill and then merge all of them to solve the compositional task. We distinguish two main classes of LoRA merging: Linear and Concatenation.

Concatenation of LoRAs. Here $\Delta W^l = \alpha_1^l B_1 A_1^\top + \alpha_2^l B_2 A_2^\top$, where $\alpha_1^l, \alpha_2^l \in [0, 1]$ are merging coefficients for layer l. We refer to this method as concatenation of LoRAs. We study two variants that differ in their merging coefficients definition:

(a) **Learnable concatenation (CAT)** (introduced in this paper): in this variant, we set α_1^l, α_2^l as trainable parameters. This distinguishes us from other methods like TIES, DARE, LoRA Hub that learn static values for every layer l of the network. Once the LoRA modules are separately trained, we add a step where we only train the merging coefficients on a small mixture of the datasets.

(b) **Mixture of Experts (MoE)** (Buehler and Buehler, 2024a; Feng et al., 2024; Luo et al., 2024; Muqeeth et al., 2024; Wu et al., 2023): the main difference of this method compared to the previous ones is that the merging coefficients are *input-dependent*. Indeed, we have a trainable router parameter $W_r^l \in \mathbb{R}^{d \times 2}$ which computes the logits $h^l(x) = W_r^{l\top} x$. Then, the merging coefficients are defined as:

$$\alpha_1^l = \frac{e^{h^l(x)_1}}{e^{h^l(x)_1} + e^{h^l(x)_2}}, \alpha_2^l = \frac{e^{h^l(x)_2}}{e^{h^l(x)_1} + e^{h^l(x)_2}}$$

Linear merging of LoRAs. Here $\Delta W^l = (\alpha_1^l B_1 + \alpha_2^l B_2)(\alpha_1^l A_1 + \alpha_2^l A_2)^\top$. We note that all these methods use the same static weights (α_1, α_2) for every layer l. Compared to the concatenation of LoRAs, this method involves additional cross-terms. We study three variants that apply a series of preprocessing steps on the LoRA parameters before applying the update.

(a) **TIES** (Yadav et al., 2024): prune the smallest values of (A_k, B_k) for $k \in \{1, 2\}$ and retain the top values based on the specified density fraction $\lambda \in [0, 1]$. Next, calculate the majority sign mask from the pruned LoRA weights by summing all their parameter values and storing the sign of this sum. Lastly, the LoRA weights are

multiplied by weights (α_1, α_2). Finally, apply the linear merging update based on the stored majority sign. $(\lambda, \alpha_1, \alpha_2)$ are hyper-parameters.

(b) **DARE** (Yu et al., 2023a): first randomly prune the values of the LoRA parameters (A_k, B_k) for $k \in \{1, 2\}$ based on a density $\lambda \in [0, 1]$. Then, rescale the pruned LoRA weights by $1/\lambda$. Finally, apply the linear merging update.

(c) **LoRA Hub** (Huang et al., 2023): this method was primarily proposed to select and assign weights to the constituent LoRA modules that would help solve an unseen test task. Here (α_1, α_2) are learned from a few (5) examples from the target task in a gradient-free manner.

Most of these aforementioned methods have been introduced in the literature for solving the multitask problem, i.e. to perform well simultaneously on k skills, when tested independently.

4 Experimental details

Our base model is Llama-7b (Touvron et al., 2023).

DATA-MIX. Training a model on a mixture of datasets in §5.3, §5.2 is not straightforward, as different examples have different masking schemes, which makes standard data mixing fail. Hence, we perform continual training by fine-tuning for 3 epochs on the chapter/manual, merge the weights with the pre-trained model weights, followed by 1 epoch of fine-tuning on the instruction-following dataset similar to Wu et al. (2024).

CAT. We freeze the trained LoRA skill modules and train α_1^l, α_2^l on a dataset made by selecting the minimum of 5% of the data points from both skill 1 and skill 2. This additional step only runs for 1 epoch with a learning rate of $1\text{e}{-4}$. Further details are discussed in §B.1 and §B.2.

5 Experiments

5.1 Hard math-word problems with code

Evaluation setup. We evaluate the ability to solve hard math-word problems using code.

Baselines. Base (Llama-7b with 8-shot PAL); Skill LoRAs: Math (trained on MetaMathQA (Yu et al., 2023b)), Code (trained on Code Alpaca (Chaudhary, 2023)). DATA-MIX (trained on [MetaMathQA ; Code Alpaca]); LoRA Merging: TIES, DARE, MoE, LoRAHub.

Results. Figure 4 illustrates that finetuning on the concatenation of MetaMathQA and Code-Alpaca – which corresponds to the DATA strategy is effective since the accuracy increases by 32% over Math (from $14.18\% \rightarrow 18.8\%$) and the model effectively exploits the synergies between natural and programming language (Xu et al., 2023). CAT is the best method; Figure 7 of Appendix shows a qualitative example comparing CAT to the next best method DATA. Despite being fine-tuned on a smaller code dataset i.e. Code Alpaca, CAT matches the performance of Code Llama - Python 7b (Roziere et al., 2023) that is specialized for Python. Additionally, CAT demonstrates *super-linear improvement* (Table 1). *Super-linear improvement* means that the set of problems we can solve with model merging is larger than the sum of the number of problems solved by the math model and the number of problems solved by the code model. This concept is independent of the absolute performance of the individual models. Correspondingly, here the improvement when fine-tuning on both math and code with respect to the base model (257%) is superior to the sum of the improvements on code only (36%) and math only (140%). For example, CAT solves 21% of the problems, meaning that at least 5% of the problems (union of both is 16%) it solves are not solved by either of the individual models. The solution to these problems must therefore arise from combining the knowledge of both models.

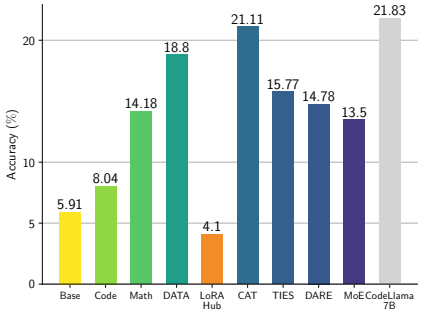


Figure 4: Performance on GSM8k-Hard.

	No Code	Code
No Math	5.91%	8.04% (+36%)
Math	14.18% (+140%)	21.11% (+257%)

Table 1: Super-linear improvement with CAT.

5.2 Building specialized question-answering bots (QABots)

Evaluation setup. We test the closed-book question-answering capability (i.e. no access to

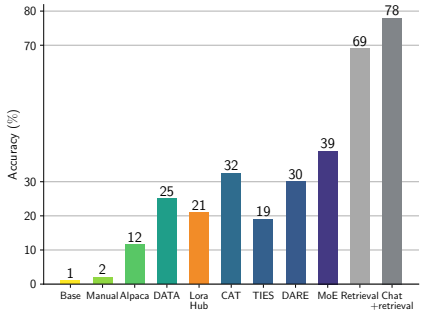
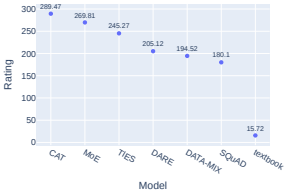
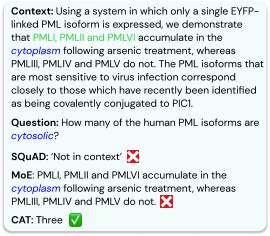


Figure 5: Performance on closed-book game QA.



(a) ELO Ratings of various models.



(b) CAT vs. MoE vs. SQuAD solving a BioASQ question.

Figure 6: Quantitative and qualitative results on reading comprehension task.

the manual about which questions are asked) by testing on the nuanced *Dungeons & Dragons* game manual. Details on the preparation of the dataset and judging are discussed in §B.3, §C.2. In contrast to subsection 5.3, this is closed-book QA as we have access to the context during inference.

Baselines. Skill LoRAs: Manual (trained on game manual), Instruction-following (trained on Alpaca). We include two additional baselines: retrieval using `langchain` based `RetrievalQAChain` using (1) `Llama-7b` and (2) `Llama2-7b-chat` models. Here the documents are embedded using `sentence-transformers/all-MiniLM-L6-v2` and stored in a FAISS vector store. These are open-book but included to get an upper bound.

Results. From Figure 5, we observe that CAT beats most merging and data mixing but we note the scope for improvement compared to the more expensive retrieval methods.

5.3 Reading comprehension on technical documents

Evaluation setup. We test the open-book question-answering ability (i.e. context is accessible to the model) on BioASQ by choosing the "factoid" subset of questions to align with the format of questions seen in SQuAD (Rajpurkar et al., 2018). Since the ground truth answers in BioASQ are very long and sometimes contain more details than what is provided in the context, we observe low results in exact matching and F1 scores (see §C.3 Table 2). To alleviate this issue, we use GPT-4 as a judge (Zheng et al., 2024): given the answers generated by a pair of models, we ask it to score the two in terms of relative correctness to the gold reference answer, and report the ELO rating (Elo and Sloan, 1978) (see §C.3 for prompts and details).

Results. Figure 6a shows that CAT obtains the best ELO rating. Evidently, the question-answering skill is more useful than having domain knowledge as SQuAD fairs 6× better compared to textbook. From Figure 6b we see the lack of biomedical knowledge hurting SQuAD; CAT gives well-formed answers compared to MoE which is indeed able to understand the context but just copies text.

5.4 Robustness to prompt format changes

Evaluation setup. We choose the task of counting the number of words having a particular letter in the given sentence from Super-NaturalInstructions (Wang et al., 2022). Following the grammar over *descriptors*, *separators*, and *spaces* defined in (Sclar et al., 2024), we sample 7 prompt formats (see Figure 3 for some examples of format variations – as simple as spacing, casing).

Results. Figure 10 reports the performance of each model when evaluating on 4 out-of-distribution formats (Formats 3, 7, 8 and 10). Figures 10a, 10b and 10c respectively display the single and merged models when trained on Formats 1&4, Formats 1&2 and Formats 2&4. DATA-MIX performs worse than the individual models. Regarding the merged models, the results are variable. When finetuning on Formats 1&4 (Figure 10a), the performance of CAT does not vary across target formats and remains high. On Formats 1&2 (Figure 10b), TIES is the best model since it performs as the single model when evaluated on Formats 3 and 10. CAT and DARE perform worse and only obtain a decent performance when evaluated on Format 3. Lastly, on Formats 2 & 4, merging fails since all the models perform poorly. Thus, we show that model merging is an approach to attaining robustness to prompt formatting changes. We study CAT for prompt robustness on other tasks in subsection C.4.

5.5 Ablations

Learning the weights of CAT. We analyze the impact of learning the weights to be assigned to each skill. As discussed in section 4, we learn the merging coefficients. Figure 9 shows how that "learned" CAT beats "static" CAT in the Math-Code and the QAbot experiments by 2% and 8% respectively. Here, we simply average – $\alpha_1^l, \alpha_2^l = 0.5$.

6 Conclusion

We conclude that when obtaining training data is challenging, decomposing a task into its underlying skills and concatenating individual skill LoRAs is a promising approach. We demonstrate several practical use cases that can be treated as such binary skill composition problems. An exciting future direction is to investigate the efficacy of CAT on tasks encompassing more than two skills. This would give an interesting alternative to the current paradigm where we train large-scale models on large data mixtures.

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*.

Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. 2024. Evolutionary optimization of model merging recipes. *Preprint*, arXiv:2403.13187.

Eric L Buehler and Markus J Buehler. 2024a. X-lora: Mixture of low-rank adapter experts, a flexible framework for large language models with applications in protein mechanics and design. *arXiv preprint arXiv:2402.07148*.

Eric L. Buehler and Markus J. Buehler. 2024b. X-lora: Mixture of low-rank adapter experts, a flexible framework for large language models with applications in protein mechanics and molecular design. *Preprint*, arXiv:2402.07148.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28:41–75.

Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation. https://github.com/sahil280114/codealpaca.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Arpad E Elo and Sam Sloan. 1978. The rating of chessplayers: Past and present. *(No Title)*.

Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Yu Han, and Hao Wang. 2024. Mixture-of-loras: An efficient multitask tuning for large language models. *arXiv preprint arXiv:2403.03432*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. 2018. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vlad Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. 2024. Arcee's mergekit: A toolkit for merging large language models. *Preprint*, arXiv:2403.13257.

Yuchao Gu, Xintao Wang, Jay Zhangjie Wu, Yujun Shi, Yunpeng Chen, Zihan Fan, Wuyou Xiao, Rui Zhao, Shuning Chang, Weijia Wu, et al. 2024. Mix-of-show: Decentralized low-rank adaptation for multi-concept customization of diffusion models. *Advances in Neural Information Processing Systems*, 36.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2023. Lorahub: Efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv:2307.13269*.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2022. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Leroo-AI. 2024. *Leeroo-AI/mergoo*.

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.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Tongxu Luo, Jiahe Lei, Fangyu Lei, Weihao Liu, Shizhu He, Jun Zhao, and Kang Liu. 2024. Moelora: Contrastive learning guided mixture of experts on parameter-efficient fine-tuning for large language models. *Preprint*, arXiv:2402.12851.

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. 2024. State of what art? a call for multi-prompt llm evaluation. *Preprint*, arXiv:2401.00595.

Mohammed Muqeeth, Haokun Liu, Yufan Liu, and Colin Raffel. 2024. Learning to route among specialized experts for zero-shot generalization. *Preprint*, arXiv:2402.05859.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *The Twelfth International Conference on Learning Representations*.

Viraj Shah, Nataniel Ruiz, Forrester Cole, Erika Lu, Svetlana Lazebnik, Yuanzhen Li, and Varun Jampani. 2023. Ziplora: Any subject in any style by effectively merging loras. *arXiv preprint arXiv:2311.13600*.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, et al. 2022. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*.

Joshua Smith and Michael Gashler. 2017. An investigation of how neural networks learn from the experiences of peers through periodic weight averaging. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 731–736. IEEE.

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

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

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

Joachim Utans. 1996. Weight averaging for neural networks and local resampling schemes. In *Proc. AAAI-96 Workshop on Integrating Multiple Learned Models. AAAI Press*, pages 133–138. Citeseer.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022. Super-naturalinstructions:generalization via declarative instructions on 1600+ tasks. In *EMNLP*.

Tom White. 2017. Sampling generative networks.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR.

Chaoyi Wu, Weixiong Lin, Xiaoman Zhang, Ya Zhang, Weidi Xie, and Yanfeng Wang. 2024. Pmc-llama: toward building open-source language models for medicine. *Journal of the American Medical Informatics Association*, page ocae045.

Xun Wu, Shaohan Huang, and Furu Wei. 2023. Mole: Mixture of lora experts. In *The Twelfth International Conference on Learning Representations*.

Yiheng Xu, Hongjin Su, Chen Xing, Boyu Mi, Qian Liu, Weijia Shi, Binyuan Hui, Fan Zhou, Yitao Liu, Tianbao Xie, et al. 2023. Lemur: Harmonizing natural language and code for language agents. *arXiv preprint arXiv:2310.06830*.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. 2024. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36.

Yang Yang, Wen Wang, Liang Peng, Chaotian Song, Yao Chen, Hengjia Li, Xiaolong Yang, Qinglin Lu, Deng Cai, Boxi Wu, and Wei Liu. 2024a. Loracomposer: Leveraging low-rank adaptation for multi-concept customization in training-free diffusion models. *Preprint*, arXiv:2403.11627.

Yang Yang, Wen Wang, Liang Peng, Chaotian Song, Yao Chen, Hengjia Li, Xiaolong Yang, Qinglin Lu, Deng Cai, Boxi Wu, et al. 2024b. Loracomposer: Leveraging low-rank adaptation for multi-concept customization in training-free diffusion models. *arXiv preprint arXiv:2403.11627*.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2023a. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *arXiv preprint arXiv:2311.03099*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023b. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

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. *Preprint*, arXiv:2402.16843.

A Review on LoRA

All the methods we study for solving composition skill problems are based on LoRA (Hu et al., 2021), which is defined as follows. During finetuning, the update of the weights are constrained to be a low-rank decomposition i.e. the update is $W_0 + \Delta W$, where $\Delta W = BA^\top$, for W_0 pre-trained weights, $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{d\times r}$ trainable parameters, and $r \ll d$.

LoRA presents several advantages compared to standard finetuning. First, it is more parameter-efficient i.e., it uses a lower number of trainable parameters and has a lower memory usage i.e., fewer parameters need to be stored and processed which lowers the memory footprint. More importantly, it is more modular i.e. LoRA's method of isolating additional parameters makes it easier to manage adaptations and switch between different fine-tuned tasks. It is possible to load and apply different sets of low-rank adaptations without needing to retrain the entire model from scratch for each new task. For these reasons, we focus on LoRA-based methods to solve skill composition problems. We detail them in the next section.

B Additional experimental details

B.1 Hyperparameters.

For skill fine-tuning, we set the LoRA rank $r = 32$, LoRA alpha $= 64$, LoRA dropout $= 0.05$ and the target modules to be `{"q_proj", "v_proj", "k_proj", "up_proj","down_proj"}`. We finetune the individual LoRAs using AdamW (Loshchilov and Hutter, 2017) for 3 epochs, with a learning rate $3\text{e}{-4}$, 100 warmup steps, a linear decaying schedule, batch size in $\{4, 8\}$ and gradient accumulation 4. We set the precision to `float16`.

The $(\lambda, \alpha_1, \alpha_2)$ hyperparameter values for TIES, DARE are chosen by doing a sweep over $\lambda \in [0, 1], \alpha_1 \in [1, 2], \alpha_2 \in [1, 2]$ in increments of 0.2 and we report the best results. At inference time, we generate answers in an autoregressive fashion setting `temperature` to 0.01, `max_new_tokens` to 200, with nucleus sampling probability `top_p` as 0.95. For LoRA Hub, we used at least 5 few-shot examples for learning the weights. We followed the implementation of *mergoo* (Leroo-AI, 2024) for MoE which is based on recent MoE for LoRAs (Feng et al., 2024; Buehler and Buehler, 2024b).

B.2 Training details & computing resources.

We run our experiments on the following GPUs depending on their availability on our compute cluster: NVIDIA RTX A6000, NVIDIA RTX A5000, and NVIDIA A100. Mainly, for the most extensive large-scale fine-tuning of five textbooks for reading comprehension, we train the models in a distributed multi-GPU environment using DeepSpeed on 2 A100 GPUs. This takes about 2 days.

For reading comprehension, we use a sequence length of 2048, while for question-answer we found that using smaller lengths in {20,100} worked better at memorizing minute details.

In all data and LoRA merging methods, we do 1-shot prompting to ensure the output is of consistent format. The best model is chosen as the one that does best on the validation set.

B.3 Datasets introduced.

For the QABot task, we create two new datasets consisting of 95 examples (45 from the biology textbook chapter 16 and 50 from the rules of *Dungeon & Dragons* manual). We extract text contents from these to prepare the training corpus. This amounts to 8169 tokens in the case of biology and 25427 tokens for the game. To obtain question-answer pairs, we provide page-level content in the context and use the Question-Answer generation prompt shown in Figure 11 to prompt GPT-4 (Achiam et al., 2023). This is followed by a manual inspection to ensure valid and appropriate questions.

C Additional experiments

C.1 Math-word problems.

Since this setting requires training over smaller scale data than reading comprehension and QABot, we conducted an evaluation to assess the robustness of the reported accuracy metric of the CAT method. We obtain 21.63 ± 1.07 when testing with 3 different seeds. which is still greater than DATA-MIX (18.8).

C.2 QABot

For judging, we use accuracy as the metric using the QA judge prompt in Figure 11 . We evaluate the QABot task on a sompler setting using a university textbook.

Biology textbook QA. In the simpler setting of a chapter from a university biology textbook, we see that the base Llama model that has been instruction fine-tuned on Alpaca is able to obtain 43%, which

Question: Gunter is trying to count the jelly beans in a jar. He asks his friends how many they think are in the jar. One says 9926805. Another says 20 more than half the first one. A third says 25% more than the first one. What is their average guess?

DATA ✗

error

```
def solution():
    guesses = [9926805, (9926805/2) + 20, (9926805/2) + 20 + 25% *
    ↪ 9926805]
    total_sum = 0
    for guess in guesses:
        total_sum += guess
        average_guess = total_sum / len(guesses)
    return average_guess
```

CAT ✓

```
def solution():
    guesses = [9926805, (9926805/2) + 20, 1.25 * 9926805]
    total = 0
    for guess in guesses:
        total += guess
        average = total/3
    return average
```

Figure 7: CAT vs. DATA solving a GSM-Hard problem. DATA makes frequent coding errors.

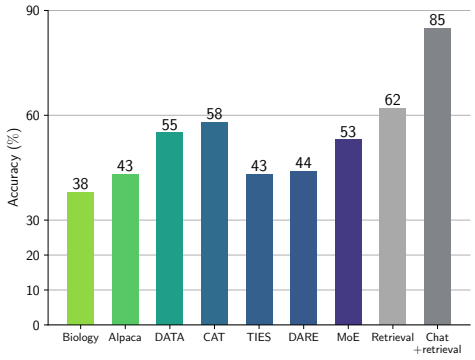


Figure 8: QABot on biology chapter.

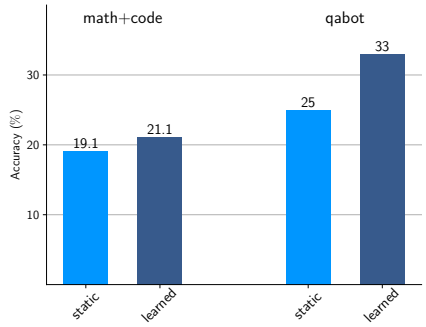


Figure 9: Performance of learned vs static CAT.

Model	F1 score
textbook (5-shot)	0.041
SQuAD	0.047
TIES	0.04
DARE	0.037
MoE	0.037
DATA	0.044
CAT	0.028

Table 2: F1 scores on BioASQ.

is enhanced by DATA-MIX which injects domain knowledge to 54%.

C.3 Reading comprehension

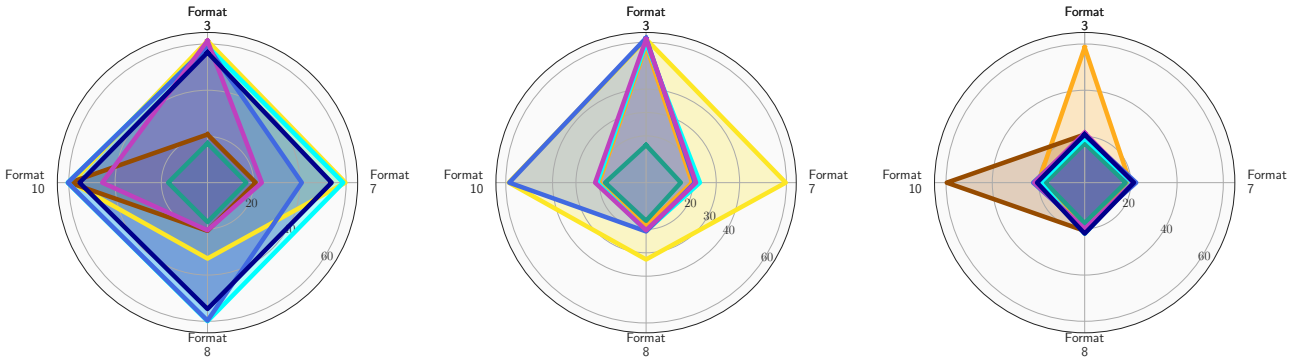
The corpus of textbooks used to impart biomedical knowledge contains 1417501 tokens.

As discussed in subsection 5.3, since the gold reference answers in BioASQ are quite descriptive unlike the simpler/concise answers for the QABot datasets, the naive F1 based scoring is unable to reflect the true performance of models Table 2. we observed that when asked to score a model individually, scores from GPT-4 do not capture the fine-grained details or consider relative generations from other models. Hence, we resort to pairwise scoring similar to LMSys. Using this scheme gives us more reliable scores. For ELO computation, we start with an initial rating of 200, base 10, scale 400, and K-factor 4. We bootstrap the ELO ratings $5,000$ times to ensure stable results.

C.4 Prompt robustness

Prompt format robustness. In Figure 10, we see different merging methods working well for different format pairs trained. While we do not see a clear strategy that would guarantee robustness, it indicates that merging methods are capable of attaining robustness. Achieving this is a very desirable phenomenon as this would eliminate the need to prompt engineer by trying diverse formatting choices.

Prompt robustness on commonsense QA tasks



(a) Format 1&4 (b) Format 1&2 (c) Format 2&4

Figure 10: Performance of single LoRAs and merged models trained on format pairs mentioned and tested on different formats.

Method	PIQA	SIQA	HellaSwag	WinoGrande
Base Llama 7B	79.8	48.9	76.1	70.1
Avg{prompt_1, prompt_2}	74.59	77.1	81.85	**81.2**
DATA-MIX	82.65	75.9	78.81	79.6
CAT	**83.27**	**78.8**	**84.71**	80.43

Table 3: Performance of CAT, DATA-MIX, and the average performance compared on 2 prompt versions of commonsense QA datasets.

Question-answer generation prompt

```
<QUESTION PROMPT>
You are an expert in generating questions. Based on the given Context, please generate a question whose answer lies in the Context. The question should be to the point and must entail a definite, concise answer in the Context. Focus on specific points and details such that only someone who understands the Context well can answer.

Context: {context}

<ANSWER PROMPT>
You are an expert at reading comprehension. Given the Context, please respond to the Question based on the information in the Context. The answer should at max be one sentence long. Output only the exact answer.
```

Reading comprehension judge prompt

```
Please act as an impartial judge and evaluate the quality of the answers.

You will receive five items: Context, Question, Gold Answer, Choice 1, and Choice 2. Your task is to assess which among Choice 1 and Choice 2 in terms of CORRECTNESS. You MUST read and understand the Context and assess the Choice answers with respect to the information in the Context. To help you with evaluation, we provide the Gold Answer. The Gold Answer has been checked by experts and is 100% correct. Use it as a reference for spotting CORRECTNESS errors.

You should give a score from 0 to 3 for CORRECTNESS to Choice 1 and Choice 2. Half points are allowed. Then you must say which choice was better.

Here is the Context:
{context}
Here is the Question:
{question}
Here is the Gold Answer:
{answer}
Here is Choice 1:
{choice1}
Here is Choice 2:
{choice2}

Please present your scores as follows:
Choice 1: x/3
Choice 2: x/3
Choice _ is better
```

Question-answer judge prompt

```
Please act as an impartial judge and evaluate the quality of the answers.

You will receive four items: Context, Question, Gold Answer, and System Answer. Your task is to assess the System Answer in terms of CORRECTNESS. You MUST read and understand the Context and assess the System Answer with respect to the information in the Context. To help you with evaluation, we provide the Gold Answer. The Gold Answer is an approximately correct answer. Use it as a reference for spotting CORRECTNESS errors but you should judge based on the provided Context.

You should give a score from 0 to 2 for CORRECTNESS to the System Answer.

Here is the Context:
{context}
Here is the Question:
{question}
Here is the Gold Answer:
{gold_answer}
Here is the System Answer:
{system_answe}
Here is Choice 2:
{choice2}

Please present your scores as follows:
Score: x
```

Figure 11: Prompts used to generate questions/judge answers using GPT-4.