# When Does LoRA Reuse Work? Theoretical Limits and Mechanisms for Recycling LoRAs Without Data Access

**Anonymous authors**
**Paper under double-blind review**

## Abstract

Reusing low-rank adapters (LoRAs) by merging or routing is a common strategy for adapting large language models to new tasks, especially when training data is unavailable but many fine-tuned LoRAs are accessible. While the availability of publicly shared LoRA weights has inspired new algorithms for composing them to solve new tasks, recent findings highlight limitations in LoRA's ability to integrate new knowledge. This work investigates when LoRA reuse could be viable without direct access to training data. Through theoretical analysis and experiments on synthetic two-hop reasoning and math word problems, we show that data-agnostic methods, such as parameter averaging and dynamic selection, often fail to combine knowledge from logically disjoint fine-tuning datasets. This challenge is particularly pronounced when the relevant knowledge is underrepresented during pretraining. However, reuse can succeed when fine-tuning datasets share solution templates, such as reasoning patterns or reusable code, which serve as bridges among tasks. Our results suggest that LoRA reuse relies more on shallow pattern matching than on logical integration of existing knowledge. This mechanism-based perspective offers practical guidance for curating datasets and designing systems that enable LoRA reuse to overcome data-access limitations. Findings indicate that future research should focus on the mechanisms enabling effective adapter reuse rather than solely on developing new reuse algorithms.

## 1 Introduction

Many real-world applications require large language models to integrate scattered information and infer logical answers to novel questions. For instance, an AI assistant supporting human resource specialists in determining an employee's tax rate must combine information about the employee's marital status and the spouse's residency, as this affects the application of tax law. Such information often originates from scarce data sources or resides in separate systems where regulatory constraints limit direct access. Hence, compositional generalization, the ability of a model to create new combinations of known elements, is essential for the quality of such applications.

As pretraining and fine-tuning become standard practices in Large Language Model (LLM) development, reusing shared model weights from foundation models and their fine-tuned variants has emerged as a practical strategy for generalization in data-scarce scenarios. Unlike Federated Learning (McMahan et al., 2017), this so-called model merging (Raffel, 2023) approach passively operates on shared model weights without coordinated training rounds. As parameter-efficient fine-tuning (PEFT) methods gain popularity, combining fine-tuned modules, especially low-rank adapters (LoRAs) (Hu et al., 2021), has emerged as a data-free alternative to enhance model capabilities (Beck et al., 2022; Huang et al., 2024; Zhao et al., 2024b; Ostapenko et al., 2024; Prabhakar et al., 2024; Zhao et al., 2024a; Yadav et al., 2025). The idea is that users can exchange and merge LoRA updates at inference time, just like plug-and-play libraries in a software program. Consequently, this idea has sparked a proliferation of novel methods for reusing fine-tuned LoRA weights for new tasks (e.g. Huang et al., 2024; Ostapenko et al., 2024; Zhao et al., 2024b; Beck et al., 2022; Zhang et al., 2025).

These approaches are appreciated for their computational and economic efficiency. However, they are often developed and validated under varying experimental conditions, with differing assumptions about system architecture, data availability, usage scenarios, and computational budgets. While recent work has addressed such inconsistencies in combining entire fine-tuned foundation models (Tam et al., 2024), the various design choices for merging or routing LoRA modules have only been surveyed (Yadav et al., 2025), leaving many questions unanswered. Furthermore, Large Language Models (LLMs) gain knowledge through pretraining, while supervised fine-tuning of instruction-following tasks teaches them the style or format for user interaction (Zhou et al., 2023). Consequently, fine-tuning LLMs with new knowledge often leads to hallucinations (Gekhman et al., 2024; Ghosal et al., 2024). Low-Rank Adaptations (LoRAs) are inherently limited in their expressiveness (Zeng & Lee, 2024) and can reduce chain-of-thought (CoT) reasoning abilities (Lobo et al., 2025).

In many real-world scenarios, producing high-quality answers is essential, yet data access is often limited (e.g., health-care, finance, medicine). **This raises our research question: how effectively can openly shared LoRA modules be reused to generate valid answers for new tasks without direct access to the corresponding raw data?** To address this, we began with the theoretical analysis revealed its limit. This is further supported by empirical findings using LLMs (parameters size ranging from 1.5B to 70B) and synthetic tasks that were designed as proxies to real-world use cases: 2-hop reasoning to combine personal information and math-reasoning in math word tasks. Together, our methods isolate factors affecting LoRA reuse, highlight challenges, and provide practical data preparation insights for designing effective solutions in those scenarios. Our key findings are:

- LoRA reuse is ineffective for new tasks regardless of model size or pretraining corpus.
- LoRA reuse can be successful without direct data-access when the common solution templates (e.g., chain-of-thoughts reasoning patterns or re-usable code snippets) existed in the fine-tuning datasets as bridges.

Overall, our findings suggest that LoRA reuse reflects shallow pattern matching rather than logically combining disjoint information among openly shared LoRAs. Understanding these mechanisms is crucial for designing systems that can effectively reuse LoRAs and create suitable fine-tuning datasets. Practitioners must consider the specific applications for composing existing LoRAs into new solutions, as curated training data is vital for successful combination.

In the following sections, we begin with a discussion of related work and overlooked perspectives. We then present theoretical analysis and empirical results that reveal the limitations of combining LoRAs in synthetic two-hop reasoning and math problem setups. Overall, our work clarifies when LoRA reuse can be effective in data-constrained settings and stress the need for rigorous, mechanism-driven studies to guide future research and practical system design.

## 2 Related Work

LoRA modules (Hu et al., 2021) have emerged as a privacy-friendly, data-free method for sharing model capabilities, allowing users to exchange LoRA updates and merge them at inference time like plug-and-play libraries (Beck et al., 2022; Huang et al., 2024; Zhao et al., 2024b; Ostapenko et al., 2024; Prabhakar et al., 2024; Zhao et al., 2024a; Yadav et al., 2025). However, many recycling methods require examples from unseen tasks to estimate merging weights or routers, raising questions about how much successful generalization can be attributed to the LoRAs themselves. This highlights the need for mechanisms ensuring effective LoRA combination under limited data access.

Weight averaging and routing are two common strategies for reusing publicly shared LoRA weights that were trained with the original PEFT approach (Hu et al., 2021). Weight averaging is a popular approach to reuse LoRAs that has been inspired by the following findings. First, fine-tuned models remain in the same loss basin as pretrained weights (Neyshabur et al., 2020). The difference between pretrained and fine-tuned model weights, or task vectors, can further steer model behavior through arithmetic operations (Ilharco et al.,

2023). Recent algorithms focus on resolving merge conflicts (TIE-Merging (Yadav et al., 2023)), randomly pruning redundant parameters (Yu et al., 2024), and estimating weights for averaging LoRAs (e.g., LoRA-hub (Huang et al., 2024)). However, the mechanisms that enable successful generalization remain unexplored. Routing is another popular alternative that has inspired by the Mixture of Experts (MoE) architecture. This type of approach treats openly shared LoRA adapters as domain experts and develops algorithms for reuse them in novel domains. However, many methods require additional data access to setup the MoE for retrieving experts or training routers (Chronopoulou et al., 2023; Zhao et al., 2024a; Jang et al., 2023). Arrow (Ostapenko et al., 2024) is a notable exception, which routes LoRAs directly based on similarity between query tokens and singular values of LoRA experts.

Beyond the standard LoRA (Hu et al., 2021), recent work has explored alternative PEFT fine-tuning and system designs to derive compositional strategies. For instance, LoRI (Zhang et al., 2025) reduces interference by freezing projection matrices and applying task-specific masks. Similarly, LoRA Soups (CAT) (Prabhakar et al., 2024) freezes one low-rank matrix during fine-tuning and subsequently merges adapters by concatenating their weights that explicitly leveraging held-out data to train a routing mechanism for the composite tasks. LoRA Lego (Zhao et al., 2024b) decomposes adapters into rank-level units for flexible recombination. Self-MoE (Kang et al., 2024) takes a different route, creating self-specialized experts with synthetic data and dynamic routing for task-specific activation.

**What can be recycled from a hub of LoRAs?** Despite the surge of novel algorithms, when privacy is paramount, choices remain limited, especially for zero-shot generalization without access to training data. The latent logic in pretraining corpus or term frequency may play roles in combining LoRAs for zero-shot generalization. Scaling language models has shown emergent abilities for zero-shot reasoning (Wei et al., 2022; Kojima et al., 2022) that suggests LLMs learned latent logical knowledge during pretraining. Task vectors demonstrate analogical reasoning through arithmetic operations (Ilharco et al., 2023), but their effectiveness may depend on term co-occurrence frequency in pretraining data (Merullo et al., 2025). If LoRAs serve as linear approximations of fine-tuned tasks, then term-frequency effects from pretraining may constrain their ability to generalize to tasks that are underrepresented in the pretraining dataset. An alternative is that observed generalization performance via merging or routing LoRAs reflects superficial pattern-matching rather than genuine compositionality. Empirical studies indicate LLMs rely on token-level cues, with small lexical changes affecting reasoning performance (Mirzadeh et al., 2024; Li et al., 2024a). LLMs struggle with latent multi-hop reasoning, relying on explicit prompting to bridge the logic gaps (Press et al., 2023; Balesni et al., 2025). Synthetic reasoning tasks thus play a key role in assessing compositional generalization, which indicates how effectively LoRA combination transfers to entirely novel tasks.

**When does reusing fine-tuned adapters work without training data?** This challenge is particularly relevant in domains such as healthcare, finance, and medicine, where data sharing is restricted but accuracy is essential. To ensure practical relevance, our study focuses on the standard LoRA (Hu et al., 2021), which is the predominant PEFT method and serves as the default in numerous open-source frameworks and cloud platforms. Our investigation begins with theoretical analysis of two-hop reasoning that shows the limits of reusing LoRAs weights. It follows with experiments using synthetic data as proxies for real-world scenarios under data constraints. The experiments were conducted on LLMs with diverse pretraining histories and parameter scales (1.5B–70B) to identify conditions for successful LoRA reuse. Finally, we extend the study to math word problems to verify how data augmentation strategies (e.g., chain-of-thought templates, reusable code snippets via multi-agent generation) and layer-targeted fine-tuning can further facilitate success. The next section details our methodology and experimental design.

## 3 Theoretical Analysis

Here, we argue theoretically that low-rank adaptation, while it can store new facts in transformers, is unlikely to lead to compositional behavior when combining different LoRAs. We study this by considering the problem of composing knowledge from two LoRAs, where each contains factual knowledge, and their combination is expected to perform two-hop reasoning (e.g. Yang et al., 2024b; Balesni et al., 2025) that requires both pieces of knowledge. In general, direct theoretical understanding of multi-layer softmax transformers is very difficult; but many theoretical insights have been obtained by studying one-layer models and the limit of

very wide models. We use this approach to perform a simple analysis of low-rank adaptation for factual knowledge. Our setup is inspired by a prior theoretical study of factual recall in transformers (Nichani et al., 2025) focusing on one-layer transformers. For simplicity, we focus on the special case of a single attention head, and do not assume noise tokens in the context.

One-layer transformers are a popular model in the theoretical literature, being more tractable than multi-layer models while offering useful insight (e.g. Peng et al., 2024; Li et al., 2024b; Sanford et al., 2023). Our empirical results (Section 4.1) confirm the applicability of our key conclusions to real-world LLMs.

Nichani et al. (2025, Section 4) show that either an MLP or an attention head can perform factual recall. Adapting the MLP with LoRA on a one-hop prompt can change individual facts – such as setting $r_1(x_1)$ to $x_2$. However, importantly, combining LoRAs adapting two relations will not result in compositional behavior, as we show below.

### 3.1 Setup

**Entities, Facts, and Prompts** We consider a set $\mathcal{X}$ of entities, and a set $\mathcal{R}$ of binary relations $r \subset \mathcal{X} \times \mathcal{X}$ (e.g., X is married to Y; Y lives in Z, etc). We assume that each $r$ is a partial function, i.e., for each $x$, there is at most one $y$ satisfying $(x, y) \in r$; we write $y = r(x)$. Whereas Nichani et al. (2025) relied on the assumption that each relation maps to a disjoint output space, we avoid this assumption. We assume that the model operates on the following types of prompts

1. One-Hop: `X REL` (where `X` represents an entity $x \in \mathcal{X}$ and `REL` represents a relation $r \in \mathcal{R}$, with expected completion: `Y`, where $y = r(x)$ (e.g., "the spouse of X is Y").

2. Two-Hop: `X REL1 REL2` (where `REL1`, `REL2` represent relations $r_1, r_2 \in \mathcal{R}$), with expected completion: `Y`, where $y = r_2(r_1(x))$ (e.g. "the place of birth of the spouse of X is Y").

**Simple Transformer Model** We consider a vocabulary consisting of relations $r_1, r_2, \ldots$ and entities $x_1, x_2, \ldots$; with token embeddings $e_{r_i}, e_{x_i} \in \mathbb{R}^d$. We will write $E \in \mathbb{R}^{|\mathcal{X} \cup \mathcal{R}| \times d}$ for the matrix holding all token embeddings. We assume a single softmax attention head with $K, Q, V \in \mathbb{R}^{d \times d}$ matrices, and a ReLU MLP with hidden dimension $m$ given by matrices $U \in \mathbb{R}^{m \times d}$; $W \in \mathbb{R}^{|\mathcal{X}| \times m}$, mapping a vector $x$ to $W \cdot \phi_{ReLU}(Ux)$. We do not require positional encodings. We assume that the next-token prediction is provided by $W$ as a one-hot encoding of the target entity, $i_x$, omitting softmax for simplicity. Our aim is to showcase limitations in composition, not in storage of knowledge itself; hence, we allow the model a width $d$ substantially larger than $|\mathcal{X}|, |\mathcal{R}|$. In order to give the MLP as much capacity as needed, we allow $m$ to be arbitrarily large.

Nichani et al. (2025) took $E$ to be randomly initialized and not trained. We follow this assumption, and additionally take $U, V$ to remain untrained, as we do not assume noise tokens in the context. Overall, we assume that $U, V, E$ matrices are randomly initialized, all with entries from $\mathcal{N}(0, \frac{1}{\sqrt{d}})$. We focus consideration of training to $W$. This represents a random features setup (e.g. Rahimi & Recht, 2008; Ghosal et al., 2022; Dirksen et al., 2022). In this setup, softmax attention is close to uniform; we will take it to be exactly uniform for simplicity. We will examine the situation where the base model already performs correctly for the given set of relations $\mathcal{R}$, and $W$ is then adapted to reflect edits to such facts.

We focus LoRA on $W$, in agreement with our experimental finding in Section 4.1 that applying to MLPs can be sufficient to get most of the gains. We consider updates $\Delta W = AB^T$ with $A \in \mathbb{R}^{|\mathcal{X}| \times s}, B \in \mathbb{R}^{m \times s}$ where $s$ is small, subject to an L2 penalty $\|A\|_F^2 + \|B\|_F^2$. We particularly consider one-rank updates, $\Delta W = pq^T$ where $p \in \mathbb{R}^{|\mathcal{X}|}, q \in \mathbb{R}^m$.

We note that this setup simplifies many aspects of transformers: there is only one layer and one head, and training focuses on the (linearized) output. We also remove the softmax over the vocabulary in the output. Our setup is designed to be *simplest possible* setup in which a nontrivial statement about LoRA's ability to learn and combine abilities can be made.

### 3.2 Results

The correct responses to all 1-hop and 2-hop relations can jointly be coded into $W$ when $d$ and $m$ are sufficiently large, due to the separation ability of the random features model (Ghosal et al., 2022). This analysis is in line with mechanistic studies of factual recall suggesting MLPs act as key-value storage (Geva et al., 2021).

Changing a fact $y = r(x)$ requires changing the output of the MLP on the subspace spanned by the entity and relation. When the update affects only a single fact, L2 regularization ensures that it has a simple and interpretable closed form:

**Proposition 1.** *A rank-one update to $W$ changing the output on a prompt* `X REL` *from $r(x)$ to $\tilde{r}(x)$ must have the form:*

$$\Delta W_{r \mapsto \tilde{r}} = \frac{1}{\|\phi_{ReLU}(UVe_{\texttt{X}} + Ue_{\texttt{REL}})\|_2^2}(i_{\tilde{r}(x)} - i_{r(x)})\phi_{ReLU}(U \cdot Ve_{\texttt{X}} + U \cdot e_{\texttt{REL}})^T \tag{1}$$

This is similar to the RoME update (Meng et al., 2022). Intuitively, based on the idea that MLPs act as key-value storage, the LoRA update $\Delta W = AB^T$ specifically addresses the encoding of the prompt `X REL` in the $B$ matrix, and the changed output in the $A$ matrix. The proof is in Appendix A.1.

Now consider a two-hop prompt `X REL1 REL2`, intended to denote the composition of the two relations. Given sufficient width, any set of such two-hop facts can be encoded in $W$. However, as we next show, adding two LoRAs modifying two relations $(\Delta W_{r_1 \mapsto \tilde{r}_1}, \Delta W_{r_2 \mapsto \tilde{r}_2})$ will not unlock compositional behavior on the new facts:

**Theorem 2.** *Assume LoRAs $\Delta W_{r_1 \mapsto \tilde{r}_1}, \Delta W_{r_2 \mapsto \tilde{r}_2}$ are created to adapt two single facts for $r_1$, $r_2$. Summing these adapters will not result in correct results for composition of the two relations $r_1, r_2$.*

The formal proof is in Appendix A.1. The reasoning is as follows. As shown in Proposition 1, the two LoRAs specifically modify the MLP output on the subspaces inhabited by the activations computed on the two one-hop prompts. When the model encounters a two-hop prompt, the activations will partly overlap with the subspaces for both one-hop prompts, and the adapters will lead the model to output not the composition $r_2(r_1(x))$, but a linear mixture of two relevant entities. A natural question is whether some of the routing or weighting methods proposed in the literature resolve this; it turns out that the argument extends to those: For instance, weighted averaging of the two adapters (e.g. Prabhakar et al., 2024; Ostapenko et al., 2024) will still fail to perform compositionally when several facts are updated. Formally (see Appendix A.1.1 for the proof),

**Theorem 3.** *The same result holds for CAT (Prabhakar et al., 2024), linear merging of LoRAs (Yadav et al., 2023; Yu et al., 2024; Huang et al., 2024), Arrow routing (Ostapenko et al., 2024).*

Yet another approach might be to combine a larger library of LoRAs where some have been trained on 2-hop examples from other task pairs. One might hope that this would prime the model towards compositional behavior; however, the reasoning above still applies, and suggests that reusing LoRAs would still fail to behave compositionally (Appendix A.1.1).

One limitation of our theoretical analysis is that (in line with Nichani et al. (2025)) it applies to a single-layer transformer; our experiments test applicability of the conclusions to LLMs across scales.

## 4 Experiments

The following experiments aim to apply the above theoretical analysis in full-scale LLMs to determine under what conditions combining LoRAs enables LLMs to perform new tasks that requires logical integration of information segmented among fine-tuning datasets. We focus on two data-agnostic routing methods, Uniform averaging and Arrow routing (Ostapenko et al., 2024), which operate directly on LoRA weights (see Appendix A.2.2). We designed two synthetic tasks, two-hop reasoning and math word problems, as proxies for real-world cases that require combining personal information or mathematical reasoning under data-scarce

conditions. This helps identify key preconditions for effective LoRA routing: entity familiarity, domain-specific pretraining, and common solution templates that link segmented information across fine-tuned adapters. We further examine how these effects hold in different routing strategies and scale across base model sizes ranging from 1.5B to 70B parameters.

### 4.1 Two-Hop generalization

We investigate whether combining two LoRAs enables compositional reasoning. Building on our theoretical analysis and inspired by Balesni et al. (2025), we design a two-hop reasoning task requiring composition across linguistic variations while controlling for base model knowledge. The dataset uses a fixed structure: **First Hop** ($A \to B$, identifying the spouse of a given entity $A$), **Second Hop** ($B \to C$, identifying the residence of $B$), with the goal of inferring $A \to C$. This setup closely follows our Theorem 2, which suggests that LoRAs trained on one of the two hops each would, if combined, not unlock the indirect relationship.

We conduct three datasets varying the nature of entity names and locations while ensuring that the relational facts remain synthetic: $F$ (fake names, fake locations), where both entities and locations are synthetic (e.g., $(Zint, Frosk, Narik)$); $H$ (fake names, real locations), where names are synthetic but locations are real (e.g., $(Zint, Frosk, London)$); and $R$ (real names, real locations), where both names and locations are real, but relationships are deliberately shuffled to remain false (e.g., $(Barack\ Obama, Camila\ Alves, London)$). We refer to the first-hop ($A \to B$), second-hop ($B \to C$), and the two-hop ($A \to C$) subsets of each dataset as $F_1, F_2, F_{12}$, $H_1, H_2, H_{12}$, and $R_1, R_2, R_{12}$, respectively (see Table 5 and Appendix A.2.1 for examples and details).

Based on ablation studies (Table 8 and 9 in Appendix A.2.2), we focused on fine-tuning only the MLP layers of the following base models: Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, and Qwen2.5-14B-Instruct Qwen-team (2025), as well as DeepSeek-R1-Distill-Qwen-7B, DeepSeek-R1-Distill-Qwen-14B, and DeepSeek-R1-Distill-LLaMA-70B DeepSeek-AI-team (2025).

#### 4.1.1 Impact of base model and familiarity

**Experiment setup** For each dataset ($F$, $H$, $R$), we train four LoRA adapters (experts) LoRA 1, LoRA 2, LoRA 3 (Oracle Expert), and LoRA 4 (Mixed Two-Hop Expert) on relation $A \to B$, $B \to C$, $A \to C$, and mixed data ($A \to B$, $B \to C$), respectively. From these experts, we construct two libraries: **2-combination library** which includes LoRA 1 and LoRA 2; and **3-combination library** which includes LoRA 1, LoRA 2, and LoRA 3. We evaluate the model's ability to generalize and infer $A \to C$ relationships. We use Chain-of-Thought (CoT) prompting during testing for the 3-combination library and the 2-combination library.

**Results and analysis** As shown in Figure 1a, performance on the 3-combination library for the $H$ dataset improves with model size, while the 2-combination library remains consistently poor. Figure 1b shows that $R$ outperforms $H$, both far exceeding $F$. Notably, with only $A \to B$ and $B \to C$ adapters (2-combination library), accuracy stays below 10%, supporting Theorem 2 that composing knowledge across separate LoRAs is inherently challenging. Even when $A \to C$ is covered (3-combination library), routing does not always succeed, in particular in smaller models and the presence of of unfamiliar entities, such as fake names or cities in the $F$ dataset. These trends hold across datasets and model families (see Table 7 in Appendix A.2.2).

#### 4.1.2 Composition requires close match between testing and training prompts

So far, we found that synthesizing two-hop reasoning from two LoRAs is difficult, in agreement with our theoretical predictions. What strategies could enable composition? While a substantial amount of work has found chain-of-thoughts to support reasoning (including in the 2-hop setup when employing full-parameter finetuning, (Balesni et al., 2025)), we found that composition from the two $A \to B$ and $B \to C$ experts performs poorly even with chain-of-thought prompting. Our theoretical analysis suggests that, as LoRA adapters rely on targeting specific low-dimensional subspaces, compositional behavior can only be unlocked when the target prompts show a close formal match to prompts on which the LoRAs were trained. CoT prompting might thus be insufficient as the form of the targets mismatches the one-hop training examples

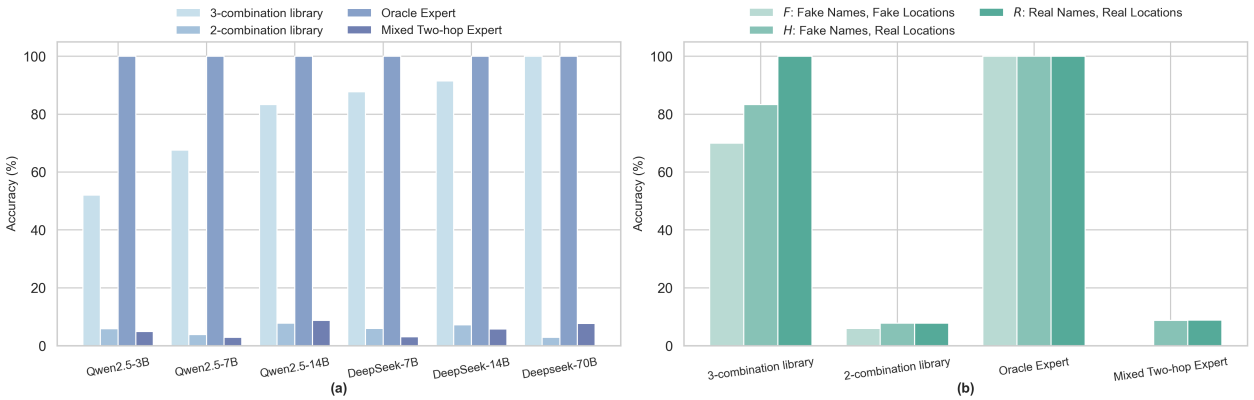


Figure 1: LoRA reuse is ineffective across model sizes and pretraining corpora, as shown by performance comparisons between a library of multiple LoRA adapters and individually fine-tuned LoRA experts on two-hop datasets. (a) Comparison of library-level and expert-level performance on the test set of $H$ across different base models. (b) Impact of entity familiarity on the performance of LoRA combination methods, using Qwen2.5-14B-Instruct as the base model, evaluated on three test sets: $F$, $H$, and $R$. Across setups and model sizes, performance on the 2-combination library is poor; including an expert trained on the target task ($A \to C$ relationship) is necessary.

of the LoRAs. However, mixing in CoT examples into the LoRA training data might be sufficient. In this section, we test if it is possible to enable composition by including CoT templates in the training data, and how close the match in reasoning patterns needs to be between finetuning and testing datasets. We specifically define the following *bridge* technique, and design a series of experiments to determine how closely the target task must be represented in the LoRA fine-tuning datasets to enable compositional behavior. The idea is that the finetuning dataset additionally includes examples of the targeted reasoning pattern. Via ablations, we test which aspects of the targeted reasoning pattern needs to be present.

**Experimental setup** We design two *bridge* variants over the $F$ and $R$ datasets. The **Fake Bridge** ($B_F$) is constructed by concatenating the $F_1$ ($A \to B$), $F_2$ ($B \to C$), and $F_{12}$ CoT (CoT-formatted $A \to C$) subsets. The **Real Bridge** ($B_R$) follows the same structure but uses the corresponding subsets ($R_1$, $R_2$, $R_{12}$ CoT) from the $R$ dataset, which contains real names and locations (see Table 6 and Figure 3 in the Appendix A.2.1 for examples and details). We fine-tune adapters on the union of direct-hop examples and a bridge dataset. In the first configuration, ***Setup 1***, models are trained by mixing fake data subsets ($F_i$) with the Real Bridge set ($B_R$): LoRA 1 is trained on $F_1 + B_R$, and LoRA 2 on $F_2 + B_R$, with evaluation on the held-out subset $F_{12}$. In the second configuration, ***Setup 2***, models are trained by mixing real data subsets ($R_i$) with the Fake Bridge set ($B_F$): LoRA 1 is trained on $R_1 + B_F$, and LoRA 2 on $R_2 + B_F$, with evaluation on the held-out subset $R_{12}$.

**Results and analysis** Figure 2 demonstrates that explicitly incorporating the target two-hop reasoning pattern into the LoRA fine-tuning data is crucial for achieving reliable compositional generalization. In ***Setups 1*** and ***2***, where each adapter is trained on a synthetic direct-hop subset combined with the bridge dataset, *Arrow* performance improves significantly compared to earlier experiments. Additionally, the bridge setup is much more successful in ***Setup 2***, highlighting the importance of entity familiarity for effective generalization.

We conducted a set of ablations to analyze which aspects are important for the success of this strategy. Incorporating structured reasoning into LoRA finetuning yields only marginal gains unless the finetuning data closely mirror the target two-hop task. First, as shown in Table 1, the bridge improves two-hop accuracy only when CoT-formatted $A \to C$ instances are included during adapter training. Omitting CoT formatting results in worse performance (***Setups 2*** vs. ***3***). Second, simply including the bridge in only one of the two LoRA adapters (***Setups 4, 5***), or providing only $A \to C$ prompts without the individual one-hop tasks

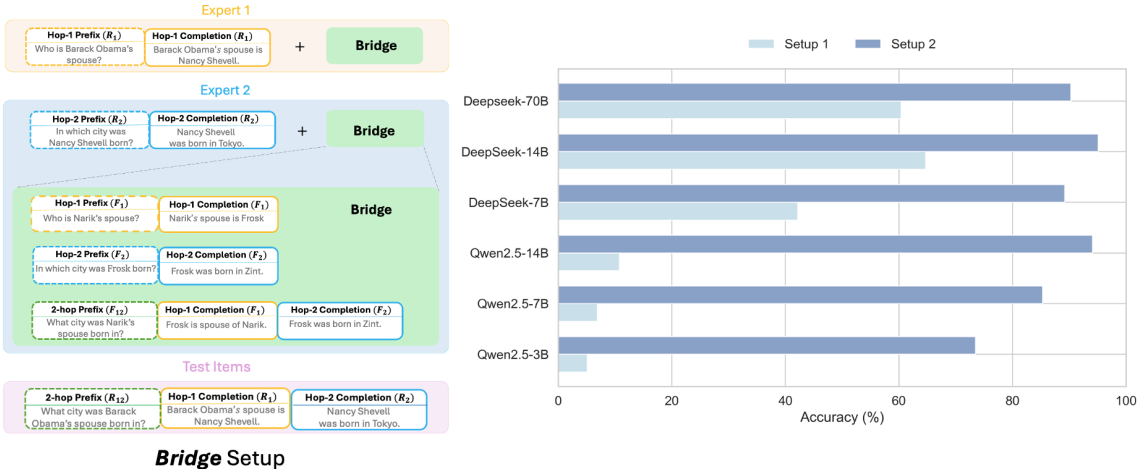


Figure 2: (left) In the bridge setup, both LoRA experts are trained not only on one of the two hops, but also on examples (disjoint from those needed in testing) of both hops, and chain-of-thought two-hop reasoning. (right) Performance comparison of two setups across different base models: Setup 1 (**Real Bridge**, $B_R$) adds a bridge using real names and locations to a dataset using fake names and fake locations ($F$), while Setup 2 (**Fake Bridge**, $B_F$) reverses these. Each setup uses LoRA 1 and LoRA 2 as the experts in the library. The bridge setup is much more successful in Setup 2.

(***Setups 6, 7***), results in significantly weaker compositional performance compared to setups where both $F_1$ ($A \to B$) and $F_2$ ($B \to C$) are included alongside CoT bridging. This highlights the importance of exposing the model to each subtask. Third, we found that relaxing the bridge to use disjoint task pairs still produces nontrivial gains (***Setup 8***, which uses completely different relations as the bridge dataset: $F_4$: *study_in* and $F_5$: *child_of*, see Table 5 in the Appendix A.2.1 for examples), suggesting that exact task-pair matching is less critical so long as the finetuning set contains examples reflecting the overall reasoning pattern. Altogether, these results confirm the importance of CoT exemplars and the individual tasks to unlocking generalization of Arrow routing, even if the examplars are semantically different from the target task.

Aside from combinations of two or three LoRAs, we further tested what happens when increasing the number of tasks present in the collection of LoRAs, including both various one-hop tasks, and also various two-hop tasks. Even in this case, we found that composition was very difficult, again in agreement with our theoretical predictions (Analysis in Appendix A.3).

Table 1: Ablations for the bridge training setups. We compare the 2-combination libraries trained on just the two hops (0), the full bridge (2), with various strategies interpolating between these, such as providing a bridge in only one expert (4 and 5), or omitting the CoT template from the bridge (3). The full bridge attains highest performance, and versions not including a bridge CoT in both of the two experts show poor performance (0, 3, 4, 5).

| Setup | LoRA 1 | LoRA 2 | Qwen2.5-3B-Instruct | Qwen2.5-7B-Instruct | Qwen2.5-14B-Instruct | DeepSeek-R1-Distill-Qwen-7B | DeepSeek-R1-Distill-Qwen-14B | DeepSeek-R1-Distill-Llama-70B |
|---|---|---|---|---|---|---|---|---|
| 0 | $R_1$ | $R_2$ | 1% | 3.9% | 7.8% | 8.8% | 3.9% | 5.9% |
| 2 | $R_1$<br>$F_1 + F_2 + F_{12}$ CoT ($B_F$) | $R_2$<br>$F_1 + F_2 + F_{12}$ CoT ($B_F$) | **73.5%** | **85.3%** | **94.1%** | **89.2%** | **95.1%** | **90.3%** |
| 3 | $R_1$<br>$F_1 + F_2 + F_{12}$ | $R_2$<br>$F_1 + F_2 + F_{12}$ | 1% | 2.9% | 3.9% | 3.9% | 11.8% | 3.9% |
| 4 | $R_1$<br>$F_1 + F_2 + F_{12}$ CoT ($B_F$) | $R_2$ | 9.8% | 8.8% | 11.8% | 8.8% | 12.7% | 10.2% |
| 5 | $R_1$ | $R_2$<br>$F_1 + F_2 + F_{12}$ CoT ($B_F$) | 10.8% | 35.3% | 27.5% | 25.9% | 22.5% | 26.1% |
| 6 | $R_1$<br>$F_{12}$ CoT | $R_2$<br>$F_{12}$ CoT | 40.2% | 61.8% | 65.7% | 75.5% | 72.5% | 87.3% |
| 7 | $R_1$<br>$F_1 + F_{12}$ CoT | $R_2$<br>$F_2 + F_{12}$ CoT | 27.5% | 75.5% | 79.4% | 74.5% | 76.9% | 89.9% |
| 8 | $R_1$<br>$F_1 + F_4 + F_{14}$ CoT | $R_2$<br>$F_5 + F_2 + F_{52}$ CoT | 43.1% | 82.4% | 92.2% | 81.6% | 83.5% | 87.9% |

We also investigate on how the 2-hop reasoning gap can be closed without reasoning templates as bridge but alternative merging methods or extra training when data access would be allowed. To achieve this, we utilize LoRA 1 and LoRA 2 in Setup 0 as shown in the Table 1 and apply uniform averaging (Huang et al., 2024; Ostapenko et al., 2024), TIES-Merging (Yadav et al., 2023) and LoRA Soups (CAT) (Prabhakar et al., 2024) on them as comparison baselines because they represent widely discussed strategies for merging openly shared fine-tuned LoRA weights. TIES-Merging applies a structured procedure—trimming low-impact parameters, resolving sign conflicts, and selectively merging—to reduce interference when combining multiple model weights. CAT altered the original LoRA (Hu et al., 2021) by only allowed one low-ranked matrices to be fine-tuned, concatenation multiple LoRA model weights, and requires access to held-out data for routing training to composite multiple tasks. Table 2 shows that when held-out data is available for routing training, CAT achieves higher accuracy than methods operating without data access. However, its performance remains below setups that route openly shared LoRAs originally fine-tuned on well-designed datasets (Table 1, setup 2,6,8).

Table 2: Performance of routing methods using a 2-combination LoRA adapter library on Qwen2.5-7B-Instruct with the $R$ dataset (Setup 0 from Table 1). CAT was trained on half of $R_{1,2}$ and evaluated on the other half, unlike the other methods that were tested without further training.

| Routing | Accuracy |
|---|---|
| uniform | 3% |
| arrow | 3.9% |
| TIES | 3.8% |
| CAT | **21%** |

Overall, these results support our conclusion that (i) direct composition of knowledge from different LoRAs is very difficult, (ii) the LoRA finetuning datasets must contain examples closely matching the target reasoning behavior.

### 4.2 Generalization from Easy to Hard Math Word Problems

To evaluate whether our findings hold in more realistic settings, we use the GSM-Symbolic benchmark (Mirzadeh et al., 2024), which enables controlled assessment of reasoning robustness in math across well-defined difficulty levels. Each LoRA expert was fine-tuned on GSM-Symbolic (original) and GSM-P1 (with one added clause) individually, before being combined for evaluation on GSM-P2 (which adds another clause). We compare general-purpose and math-specialized models to assess the impact of pretraining. Similar to exposing LoRAs to solutions closely resembling the target task, we also tested whether fine-tuning with reusable Markdown and Python code (Suzgun et al., 2025) would improve generalization on GSM-P2. Detailed experimental design, fine-tuning, and evaluation procedures can be found in Appendix Section A.4.1.

**Limitations of LoRA reuse.** The effectiveness of LoRA routing is highly dependent on the base model's domain-specific pretraining history. To start, we replicated the findings of Mirzadeh et al. (2024), which show that large language models (LLMs) lack robustness in mathematical reasoning (see Appendix Section A.4.2, Table 12). Routing methods such as Uniform and Arrow provided modest improvements for the general-purpose Qwen2.5-1.5B-Instruction model, but often degraded performance in math-specialized models like Qwen2.5-Math-Instruction, regardless of model size (Table 3). Among these, Uniform consistently outperformed Arrow. Echoing prior work showing that 8-shot GSM8K in-context examples do not improve performance on GSM-P2 (Mirzadeh et al., 2024), we further observed that combining these examples with LoRA routing actually worsened results. For example, in the Qwen2.5-Math-7B-Instruction model, Arrow routing with in-context examples reduced GSM-P2 accuracy from 0.27 to 0.06 (see Appendix Section A.4.2, Table 13 for details).

The performance drop observed after LoRA routing may stem from a mismatch between the fine-tuning data and the base model's capabilities. Qwen2.5-Math-Instruction is designed to solve problems using Markdown and Python code, while the GSM-Symbolic benchmarks provide only natural language Chain-of-Thought (CoT) solutions. As a result, routing LoRAs fine-tuned on this dataset may suppress the model's tool-integrated reasoning abilities and lead to an increase in calculation errors. Our error analysis follows the

definitions and procedures outlined by Zhong et al. (2025). See Appendix Section A.4.2 and Table 14 for details.

Table 3: Accuracy comparison on zero-shot GSM-P2 after routing LoRA experts individually fine-tuned on GSM-Symbolic and GSM-P1.

| **LoRA Routing Methods** | **Qwen2.5-1.5B-Inst** | **Qwen2.5-Math-1.5B-Inst** | **Qwen2.5-Math-7B-Inst** |
|---|---|---|---|
| Base model only | 5% | 47% | 68% |
| Uniform | 10% | 24% | 34% |
| Arrow | 9% | 19% | 27% |

**How can programming language bridge the generalization gap?** This experiment extends our bridge results in two-hop reasoning (Table 1) to math word problems. The experimental design is motivated by recent findings, Dynamic Cheatsheet, which demonstrate that encouraging language models to retain and apply reusable intermediate solutions during inference significantly improves their performance on math problems (Suzgun et al., 2025). We extend this idea using the GSM-Symbolic benchmark (Mirzadeh et al., 2024), where generalization from easier to harder problem variants requires understanding the full computational graph (Appendix Figure 5). In the previous setting, each LoRA is fine-tuned on partial solutions corresponding to subsets of reasoning steps (e.g., the black or orange subgraphs in Appendix Figure 5). However, routing these LoRAs alone does not suffice to solve the more complex P2 variant, which involves the complete computational graph (blue subgraph in Appendix Figure 5). We hypothesize that reusable Markdown and Python solutions can bridge partial representations and enhance compositional generalization through LoRA routing. To test this, we implemented two agent-based actor-critic workflows (Wu et al., 2024) to automatically generate reusable code snippets as fine-tuning data (See Appendix A.4.1 for implementation details). Table 4 demonstrate modest improvements in solving the complex P2 problems via routing LoRAs fine-tuned with these reusable code snippets. Such improvement is clearer in smaller model (Qwen2.5-Math-1.5B-Instruction) when fine-tuning targeted the MLP layers. This result aligns with our theoretical analysis and findings from the two-hop reasoning task. It highlights the importance of the mechanistic understanding of how to reuse LoRA adapters effectively for guiding data generation, and shows that such reuse works best when target tasks are clearly defined in advance.

Table 4: Enhancing easy-to-hard generalization by leveraging Tool-Integrated Reasoning (TIR) prompt and fine-tuning with reusable code.

| **Base model** | **Fine-tuned Modules** | **Routing Methods** | **LoRA(GSM-Symb) and LoRA(GSM-Symb-P1)** | **Base Model Only** |
|---|---|---|---|---|
| Qwen2.5-Math-1.5B-Inst | attention | Uniform | 14% | 18% |
| | | Arrow | 14% | |
| | MLP | Uniform | **20%** | |
| | | Arrow | 13% | |
| Qwen2.5-Math-7B-Inst | attention | Uniform | **40%** | 47% |
| | | Arrow | 24% | |
| | MLP | Uniform | 39% | |
| | | Arrow | 37% | |

## 5 Discussion

Our findings indicate that combining LoRAs is ineffective for new tasks unless common solutions to those tasks are already represented in the fine-tuning datasets. Although alternative PEFT methods may offer better compositional results, our study focuses on merging adapters without access to further training data. It is because allowing training would enable reconfiguration of adapters and task setups that make the reuse problem trivial. While comprehensive empirical comparison is beyond our scope, we discuss them using our theoretical and empirical insights for guiding the future directions. For instance, LoRI (Zhang et al., 2025) addresses cross-task inference by combining random projections with task-specific masks, potentially enabling better adapter routing. However, positive results for compositional reasoning have not been reported, and

our theoretical analysis suggests that it remains challenging. Similarly, LoRA Lego (Zhao et al., 2024b) formalizes low-rank updates as composed of independent units and clusters these into new adapters to reduce interference, though it has not been shown to enable compositional reasoning. Self-MoE (Kang et al., 2024) constructs experts based on self-generated specialization training data and a trained router, but it remains underexplored to what extent this method can enable compositional combination of different abilities. FLiX (Sun et al., 2024) learns different low-rank updates for various task or dataset features, and CS-ReFT (Sun et al., 2024) learns orthonormal task-specific adaptors. Despite these innovations, none have demonstrated effective compositional combination of skills, as our theoretical analysis suggests inherent limitations.

Recent work (Prabhakar et al., 2024) has proposed LoRA concatenation as an effective method for composing skills to solve challenging math word problems, such as those in GSM-Hard (Gao et al., 2023). We recognize the significance of these findings, particularly their demonstration that decomposing skills into reusable LoRAs and estimating appropriate combination weights can enhance performance, provided that additional task-specific data and knowledge are available. However, our work takes a different perspective. Unlike GSM-Hard (Gao et al., 2023), which primarily modifies numerical ranges while preserving the question format of the original GSM8K problems, GSM-Symbolic-P2 (Mirzadeh et al., 2024) presents more realistic and difficult compositional generalization challenges. It altered the question format and the structural complexity of math problems into an entirely unseen problem forms. Our theoretical analysis shows limitations (Appendix A.1.1) supported by empirical results that training showed little gains in a 2-hop reasoning setting (Appendix Table 2). This suggests that the benefits of such approaches may not extend to more challenging generalization tasks like GSM-Symbolic. While skill composition remains important, our results highlight a key limitation of LoRA routing approaches as shown in our theoretical analyses and empirical findings: their effectiveness often depends on prior knowledge or training data of the downstream tasks, which may not be viable in practice.

## 6 Conclusions

Our experiments explored the conditions under which combining LoRAs can enable LLMs to perform new tasks by integrating abilities from different LoRAs. We focused on two domains: two-hop reasoning and mathematical reasoning. In two-hop reasoning, we found that combining multiple LoRAs does not significantly improve performance unless the tasks are already represented in the fine-tuning datasets. Familiarity with entities enhances generalization, but the inherent challenge of composing knowledge from separate LoRAs remains. Incorporating chain-of-thought templates into training data provides only marginal improvements, particularly when fine-tuning MLP layers.

In mathematical reasoning, the effectiveness of LoRA merging is highly dependent on the base model's pre-training history. Models with domain-specific pretraining show better performance on complex tasks, while general-purpose models yield inconsistent results. Using reusable Python code solutions as fine-tuning datasets offers slight benefits, but the overall challenge of merging LoRAs persists.

Overall, our findings indicate that combining LoRAs is ineffective for new tasks unless those tasks are already represented in the fine-tuning datasets. Familiarity with entities and domains is crucial, but routing strategies often interfere with performance, especially in smaller models and unfamiliar contexts. Targeting MLP layers during fine-tuning may offer some advantages, but the direct composition of knowledge from different LoRAs remains problematic. These results indicate that a shift in focus from algorithmic innovation to understanding the boundaries of adapter-based merging is necessary, leveraging synthetic data and theoretical analysis to design effective systems.

## References

Francis Bach. *Learning theory from first principles.* MIT press, 2024.

Mikita Balesni, Tomek Korbak, and Owain Evans. The two-hop curse: Llms trained on a→b, b→c fail to learn a→c, 2025.

Tilman Beck, Bela Bohlender, Christina Viehmann, Vincent Hane, Yanik Adamson, Jaber Khuri, Jonas Brossmann, Jonas Pfeiffer, and Iryna Gurevych. AdapterHub playground: Simple and flexible few-shot learning with adapters. In Valerio Basile, Zornitsa Kozareva, and Sanja Stajner (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 61–75, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-demo.6.

Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. Open llm leaderboard, 2023.

Lucas Caccia, Edoardo Ponti, Zhan Su, Matheus Pereira, Nicolas Le Roux, and Alessandro Sordoni. Multi-head adapter routing for cross-task generalization, 2023.

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

Alexandra Chronopoulou, Jonas Pfeiffer, Joshua Maynez, Xinyi Wang, Sebastian Ruder, and Priyanka Agrawal. Language and task arithmetic with parameter-efficient layers for zero-shot summarization, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

DeepSeek-AI-team. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

Sjoerd Dirksen, Martin Genzel, Laurent Jacques, and Alexander Stollenwerk. The separation capacity of random neural networks. *Journal of Machine Learning Research*, 23(309):1–47, 2022.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10764–10799. PMLR, 23–29 Jul 2023.

Zorik Gekhman, Gal Yona, Roee Aharoni, Matan Eyal, Amir Feder, Roi Reichart, and Jonathan Herzig. Does fine-tuning llms on new knowledge encourage hallucinations?, 2024.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, 2021.

Gaurav Rohit Ghosal, Tatsunori Hashimoto, and Aditi Raghunathan. Understanding finetuning for factual knowledge extraction. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 15540–15558. PMLR, 21–27 Jul 2024.

Promit Ghosal, Srinath Mahankali, and Yihang Sun. Randomly initialized one-layer neural networks make data linearly separable. *arXiv preprint arXiv:2205.11716*, 2022.

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

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

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic, 2023.

Joel Jang, Seungone Kim, Seonghyeon Ye, Doyoung Kim, Lajanugen Logeswaran, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Exploring the benefits of training expert language models over instruction tuning. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Junmo Kang, Leonid Karlinsky, Hongyin Luo, Zhen Wang, Jacob Hansen, James Glass, David Cox, Rameswar Panda, Rogerio Feris, and Alan Ritter. Self-moe: Towards compositional large language models with self-specialized experts. *arXiv preprint arXiv:2406.12034*, 2024.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pp. 22199–22213, 2022.

Nicolas Le Roux and Yoshua Bengio. Continuous neural networks. In *Artificial Intelligence and Statistics*, pp. 404–411. PMLR, 2007.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers, 2024a.

Zihao Li, Yuan Cao, Cheng Gao, Yihan He, Han Liu, Jason Klusowski, Jianqing Fan, and Mengdi Wang. One-layer transformer provably learns one-nearest neighbor in context. *Advances in Neural Information Processing Systems*, 37:82166–82204, 2024b.

Elita Lobo, Chirag Agarwal, and Himabindu Lakkaraju. On the impact of fine-tuning on chain-of-thought reasoning, 2025.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282. PMLR, 20–22 Apr 2017.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

Jack Merullo, Noah A. Smith, Sarah Wiegreffe, and Yanai Elazar. On linear representations and pretraining data frequency in language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models, 2024.

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 512–523. Curran Associates, Inc., 2020.

Eshaan Nichani, Jason D. Lee, and Alberto Bietti. Understanding factual recall in transformers via associative memories. In *The Thirteenth International Conference on Learning Representations*, 2025.

Oleksiy Ostapenko, Zhan Su, Edoardo Ponti, Laurent Charlin, Nicolas Le Roux, Lucas Caccia, and Alessandro Sordoni. Towards modular LLMs by building and reusing a library of LoRAs. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 38885–38904. PMLR, 21–27 Jul 2024.

Binghui Peng, Srini Narayanan, and Christos Papadimitriou. On limitations of the transformer architecture. In *First Conference on Language Modeling*, 2024.

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

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.378.

Qwen-team. Qwen2.5 technical report, 2025.

Colin Raffel. Building machine learning models like open source software. *Commun. ACM*, 66(2):38–40, January 2023. ISSN 0001-0782. doi: 10.1145/3545111.

Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. *Advances in neural information processing systems*, 21, 2008.

Clayton Sanford, Daniel J Hsu, and Matus Telgarsky. Representational strengths and limitations of transformers. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 36677–36707. Curran Associates, Inc., 2023. URL `https://proceedings.neurips.cc/paper_files/paper/2023/file/73bf692447f174984f30499ec9b20e04-Paper-Conference.pdf`.

Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. Improving loRA in privacy-preserving federated learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory, 2025.

Derek Tam, Yash Kant, Brian Lester, Igor Gilitschenski, and Colin Raffel. Realistic evaluation of model merging for compositional generalization, 2024.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022. doi: 10.48550/arXiv.2206.07682.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A. Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, 2023.

Prateek Yadav, Colin Raffel, Mohammed Muqeeth, Lucas Caccia, Haokun Liu, Tianlong Chen, Mohit Bansal, Leshem Choshen, and Alessandro Sordoni. A survey on model moerging: Recycling and routing among specialized experts for collaborative learning. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. Survey Certification.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement, 2024a.

Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. Do large language models latently perform multi-hop reasoning? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10210–10229, 2024b.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 57755–57775. PMLR, 21–27 Jul 2024.

Yuchen Zeng and Kangwook Lee. The expressive power of low-rank adaptation, 2024.

Juzheng Zhang, Jiacheng You, Ashwinee Panda, and Tom Goldstein. Lori: Reducing cross-task interference in multi-task low-rank adaptation, 2025.

Ziyu Zhao, Leilei Gan, Guoyin Wang, Wangchunshu Zhou, Hongxia Yang, Kun Kuang, and Fei Wu. LoraRetriever: Input-aware LoRA retrieval and composition for mixed tasks in the wild. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 4447–4462, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.263.

Ziyu Zhao, Tao Shen, Didi Zhu, Zexi Li, Jing Su, Xuwu Wang, Kun Kuang, and Fei Wu. Merging loras like playing lego: Pushing the modularity of lora to extremes through rank-wise clustering, 2024b.

Qihuang Zhong, Kang Wang, Ziyang Xu, Juhua Liu, Liang Ding, and Bo Du. Achieving >97

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. Lima: Less is more for alignment. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 55006–55021. Curran Associates, Inc., 2023.

# A Technical Appendices and Supplementary Material

This appendix offers supplementary details to support our position in the main text, including:

- **A.1:** Formal theoretical analysis.
- **A.2:** Detailed experimental setups and results for 2-hop experiments.
- **A.3:** Evaluating Larger Sets of LoRAs.
- **A.4:** Detailed experimental setups and results for easy-to-hard math word problems.

## A.1 Theoretical Analysis

**Transformer Model** Here, we spell out the one-layer transformer model more formally. We consider a vocabulary consisting of relations $r_1, r_2, \ldots$ and entities $x_1, x_2, \ldots$; with token embeddings $e_{r_i}, e_{x_i} \in \mathbb{R}^d$. We will write $E \in \mathbb{R}^{|\mathcal{X} \cup \mathcal{R}| \times d}$ for the matrix holding all token embeddings.

We assume a single softmax attention head with $K, Q, V \in \mathbb{R}^{d \times d}$ matrices, and a ReLU MLP with hidden dimension $m$ given by matrices $U \in \mathbb{R}^{m \times d}$; $W \in \mathbb{R}^{|\mathcal{X}| \times m}$, mapping a vector $x$ to $W \cdot \phi_{ReLU}(Ux)$.

We assume that the $K, Q, U, V, E$ matrices are randomly initialized, all with entries from $\mathcal{N}(0, \frac{1}{\sqrt{d}})$. Only $W$ is trained.

Given a prompt

$$p_1 \ldots p_N \tag{2}$$

where each token $p_i \in \mathcal{X} \cup \mathcal{R}$, the input is embedded into token embeddings, which can be written as:

$$E_{p_i} \ldots E_{p_N} \in \mathbb{R}^d \tag{3}$$

where each $E_{p_i}$ is either an $e_r$ ($r$ a relation) or $e_x$ ($x$ an entity). We assume strict causal masking, where attention is allocated to the preceding positions; the activation at the current position is forward via a residual connection. The attention head, at the query position $N$, outputs

$$\sum_{i=1}^{N-1} \frac{\exp(E_{p_i}^T K^T Q e_{p_N})}{\sum_{j=1}^{N-1} \exp(E_{p_j}^T K^T Q e_{p_N})} V E_{p_i} \tag{4}$$

With the described initialization, the individual attention logits $E_{p_i}^T K^T Q e_{p_N}$ are close to zero when $d$ is large, i.e., attention is close to uniform. We will, going forward, take attention to be exactly uniform for analytical simplicity:

$$\frac{1}{N-1} \sum_{i=1}^{N-1} V E_{p_i} \tag{5}$$

In addition, the residual connection contributes $E_{p_N}$. Then, the ReLU MLP outputs

$$\begin{aligned} & W \cdot \phi_{ReLU}\left(U \cdot \left[E_{p_N} + \frac{1}{N-1} \sum_{i=1}^{N-1} V E_{p_i}\right]\right) \\ = & W \cdot \phi_{ReLU}\left(U E_{p_N} + \frac{1}{N-1} \sum_{i=1}^{N-1} U V E_{p_i}\right) \end{aligned}$$

**Formalizing LoRA** We formalize training of a LoRA adapter $\Delta W = AB^T$ on a training prompt (e.g. `X REL Y`) as choosing an adapter for which the model with adapted parameters $W + \Delta W$ outputs the target entity `Y` after the prompt `X REL`, while (among all such interpolating adapters) minimizing the regularizer $\|A\|_F^2 + \|B\|_F^2$.

**Formal Proofs** Here, we provide the formal proofs of our results stated in the main text.

**Proposition 4** (Repeated from Proposition 1)**.** *A rank-one update to $W$ changing the output on a prompt* `X` `REL` *from $r(x)$ to $\tilde{r}(x)$ must have the form:*

$$\Delta W_{r\mapsto\tilde{r}} = \frac{1}{\|\phi_{ReLU}(UVe_{\texttt{X}} + Ue_{\texttt{REL}})\|_2^2}(i_{\tilde{r}(x)} - i_{r(x)})\phi_{ReLU}(U \cdot Ve_{\texttt{X}} + U \cdot e_{\texttt{REL}})^T \tag{6}$$

*Proof.* We have the demand that

$$(W + \Delta W_{r\mapsto\tilde{r}})\phi_{ReLU}(U \cdot Ve_{\texttt{X}} + U \cdot e_{\texttt{REL}}) = i_{\tilde{r}(x)} \tag{7}$$

hence

$$\Delta W_{r\mapsto\tilde{r}}\phi_{ReLU}(U \cdot Ve_{\texttt{X}} + U \cdot e_{\texttt{REL}}) = i_{\tilde{r}(x)} - i_{r(x)} \tag{8}$$

Setting

$$\begin{aligned} pq^T =& \Delta W_{r\mapsto\tilde{r}} \\ v =& \phi_{ReLU}(U \cdot Ve_{\texttt{X}} + U \cdot e_{\texttt{REL}}) \\ w =& i_{\tilde{r}(x)} - i_{r(x)} \end{aligned}$$

we consider the general problem of finding $p, q$ such that

$$pq^T v = w \tag{9}$$

while minimizing the regularizer $\|p\|_2^2 + \|q\|_2^2$. First, by rearranging $p = \frac{w}{q^T v}$. We now use the regularizer to show that $q$ must be a multiple of $v$. Substituting into the objective, we find:

$$J(q) := \|p\|_2^2 + \|q\|_2^2 = \frac{\|w\|_2^2}{(q^T v)^2} + \|q\|_2^2. \tag{10}$$

with

$$\nabla J(q) = -2\,\frac{\|w\|_2^2}{(q^T v)^3}\,v \;+\; 2\,q. \tag{11}$$

Setting $\nabla J(q) = 0$ leads to

$$q = \frac{\|w\|_2^2}{(q^T v)^3}\,v.$$

Hence, $q$ is a multiple of $v$, and $p$ is a multiple of $w$; thus, for some scalar $\alpha$, $pq^T = \alpha w v^T$, and

$$w = pq^T v = \alpha w v^T v = \alpha w \|v\|_2^2 \tag{12}$$

and $\alpha = \frac{1}{\|v\|_2^2}$. The result follows.

□

**Theorem 5** (Repeated from Theorem 2)**.** *Assume LoRAs $\Delta W_{r_1\mapsto\tilde{r}_1}, \Delta W_{r_2\mapsto\tilde{r}_2}$ are created to adapt two single facts for $r_1$, $r_2$. Summing these adapters will not result in correct results for composition of the two relations $r_1, r_2$.*

*Proof.* We have

$$\begin{aligned} e_r, e_x &\in \mathbb{R}^d \\ V &\in \mathbb{R}^{d\times d} \\ U &\in \mathbb{R}^{m\times d} \end{aligned}$$

We first note that, in the regime $m \to \infty$, the random features model given by the initialization is represented by the kernel (Le Roux & Bengio (2007); Section 7.4.3 in Bach (2024))

$$k(x, x') \propto \frac{\|x\|_2 \|x'\|_2}{2\pi}((\pi - \eta)\cos\eta + \sin\eta) \tag{13}$$

where

$$\cos\eta = \frac{x^T x'}{\|x\|_2 \|x'\|_2} \tag{14}$$

where

$$k(x, x') \approx \frac{1}{m}\phi_{ReLU}(Ux)^T \phi_{ReLU}(Ux') \tag{15}$$

in the limit where $m \to \infty$. Consider

$$\tilde{r}_1(x) = y$$
$$\tilde{r}_2(y) = z$$

To understand the model prediction on a two-hop prompt

$$\texttt{X REL1 REL2} \tag{16}$$

after adding the two LoRAs, we consider

$$(W + \Delta W_{r_1 \mapsto \tilde{r}_1} + \Delta W_{r_2 \mapsto \tilde{r}_2}) \cdot \phi_{ReLU}\left(U \cdot \frac{V}{2} \cdot e_{\texttt{X}} + U \cdot \frac{V}{2} \cdot e_{REL1} + U \cdot e_{REL2}\right) \tag{17}$$

where the factors $\frac{1}{2}$ reflect the uniform attention weights of the single head that forwards information from preceding token, with value projection matrix $V$. We can write the second factor as

$$\phi_{ReLU}\left[U \cdot \underbrace{\left(\frac{V}{2} \cdot e_{\texttt{X}} + \frac{V}{2} \cdot e_{REL1} + e_{REL2}\right)}_{\xi :=}\right] \tag{18}$$

By Proposition 1,

$$\Delta W_{r_1 \mapsto \tilde{r}_1} \propto (i_y - i_{r_1(x)}) \cdot \phi_{ReLU}(U \cdot \underbrace{(V e_{\texttt{X}} + e_{REL1})}_{\eta_1 :=})^T$$
$$\Delta W_{r_2 \mapsto \tilde{r}_2} \propto (i_z - i_{r_2(y)}) \cdot \phi_{ReLU}(U \cdot \underbrace{(V e_{\texttt{Y}} + e_{REL2})}_{\eta_2 :=})^T$$

In the regime $m \to \infty$,

$$\begin{aligned}&(\Delta W_{r_1 \mapsto \tilde{r}_1} + \Delta W_{r_2 \mapsto \tilde{r}_2}) \cdot \phi_{ReLU}\left(U \cdot \frac{V}{2} \cdot e_{\texttt{X}} + U \cdot \frac{V}{2} \cdot e_{REL1} + U \cdot e_{REL2}\right)\\ \approx &(i_y - i_{r_1(x)}) \cdot \frac{k(\eta_1, \xi)}{k(\eta_1, \eta_1)} + (i_z - i_{r_2(y)}) \cdot \frac{k(\eta_2, \xi)}{k(\eta_2, \eta_2)}\end{aligned}$$

Adding the two adapters simply contributes nonzero multiples of both terms in

$$i_y - i_{r_1(x)} \text{ and } i_z - i_{r_2(y)}, \tag{19}$$

instead of the correct $i_z - i_{r_2(y)}$. As an addition, we also note that the newly added knowledge will not compositionally interact with knowledge stored for other entities. Consider a prompt

$$\texttt{U REL1 REL2} \tag{20}$$

where `U` denotes a different entity, not the entity `X`. Then the subspaces addressed by the two LoRAs would again have overlap with the activation on this prompt, and they would again contribute a linear combination of (19), even though neither $y$ nor $z$ might be relevant.

□

### A.1.1 Further Discussion

**Extensions to larger numbers of facts** The reasoning in the above result can be expanded to the setting where the LoRA adapters are trained for more than one new fact. In this case, the limitation shown by the theorem will become even more problematic: Assume that, say

$$\tilde{r}_1(x) = y$$
$$\tilde{r}_1(u) = v$$
$$\tilde{r}_2(y) = z$$
$$\tilde{r}_2(v) = w$$

In the regime where $d$ is large, each of the updates $\Delta W_{r_1 \mapsto \tilde{r}_1}$, $\Delta W_{r_2 \mapsto \tilde{r}_2}$ will be approximately a sum of rank-one updates of the form (6), with cross-terms from the overlap due to the shared encoding vector of the relation. More specifically, for

$$\rho_1 = \phi_{ReLU}(U \cdot (Ve_y + e_{REL2}))$$
$$\rho_2 = \phi_{ReLU}(U \cdot (Ve_v + e_{REL2}))$$

we have

$$\Delta W_{r_2 \mapsto \tilde{r}_2} = \begin{pmatrix} i_z - i_{r_2(y)} & i_w - i_{r_2(v)} \end{pmatrix} \begin{pmatrix} \rho_1^T \rho_1 & \rho_1^T \rho_2 \\ \rho_1^T \rho_2 & \rho_2^T \rho_2 \end{pmatrix}^{-1} \begin{pmatrix} \rho_1 & \rho_2 \end{pmatrix}^T \tag{21}$$

and analogously for $\Delta W_{r_1 \mapsto \tilde{r}_1}$. When run on a prompt

$$\texttt{X REL1 REL2} \quad \text{or} \quad \texttt{U REL1 REL2} \tag{22}$$

the adapter $\Delta W_{r_2 \mapsto \tilde{r}_2}$ will, in the regime where $m$ is large, contribute the same multiple of

$$(i_z - i_{r_2(y)}) + (i_w - i_{r_2(v)}) \tag{23}$$

in *both* cases, showing that the knowledge contributed by $\Delta W_{r_2 \mapsto \tilde{r}_2}$ fails to be compositionally integrated.

**Extension to other routing and merging methods** Another important consideration is how the argument extends to other routing or merging methods beyond summing adapters.

**Theorem 6** (Repeated from Theorem 3)**.** *The same result holds for CAT (Prabhakar et al., 2024), linear merging of LoRAs (Yadav et al., 2023; Yu et al., 2024; Huang et al., 2024), Arrow routing (Ostapenko et al., 2024).*

*Proof.* We first consider CAT (Prabhakar et al., 2024):

$$\Delta W = \alpha_1 B_1 A_1^T + \alpha_2 B_2 A_2^T \tag{24}$$

where the output on the prompt is a weighted combination, with fitted weights $\alpha_1, \alpha_2$. Increasing the weight belonging to the relation contributing the second hop would make the output correct on a specific two-hop example, but it would be "for the wrong reasons" and not generalize to a situation where compositions in both directions play role, or where the number of facts increases (as discussed above under "Extensions to larger numbers of facts"). Essentially the same argument applies to linear merging of LoRAs (Yadav et al., 2023; Yu et al., 2024; Huang et al., 2024):

$$\Delta W = (\alpha_1 B_1 + \alpha_2 B_2)(\alpha_1 A_1 + \alpha_2 A_2)^T \tag{25}$$

A special case of this, with $\alpha_1 = \alpha_2$, corresponds to Uniform routing. We next consider Arrow routing (Ostapenko et al., 2024), which determines weights $w_1$, $w_2$ based on similarity of the activations to the subspaces addressed by the two adapters:

$$\Delta W = (w_1 B_1 + w_2 B_2)(w_1 A_1 + w_2 A_2)^T \tag{26}$$

Applied in the setup of Theorem 2, this will just add weights based on $\left|\frac{k(\eta_1,\xi)}{k(\eta_1,\eta_1)}\right|$ and $\left|\frac{k(\eta_2,\xi)}{k(\eta_2,\eta_2)}\right|$, and the combined LoRAs will again just contribute multiples of the two terms in (19). Taken together, across linear methods combining LoRAs, no compositional behavior is expected. □

**Extension to larger number of adapters** Our arguments also extend to combining a large library with adapters that include both one-hop tasks and (other, different from the target) two-hop tasks. Each of these adapters will address subspaces spanned by activations of the relevant one-hop or two-hop prompts, and the output will be just a linear combination of different output entities, weighted depending on overlap with the subspace spanned by the activation computed on the test prompt, without computing the function composition.

### A.2 Detailed experimental setups and results for Two-hop experiments.

#### A.2.1 Experimental setups

To encourage generalization and capture the diversity of natural language, we manually created 50 paraphrased templates for each relational statement. Examples include: (*"Who is the partner of A?"*, *"Where is B's residence?"*, *"Where does the spouse of A live?"*) and (*"Who is A married to?"*, *"Where does B live?"*, *"Where is A's partner living?"*).

Each dataset — $F$ (fake names, fake locations), $H$ (fake names, real locations), and $R$ (real names, real locations) — contains 100 triplets $(A, B, C)$ spanning two relations ($A \to B$ and $B \to C$), paired with 50 paraphrase templates to generate 5,000 examples. We use a *template-based split*, assigning 46 templates for training, 2 for development, and 2 for testing per triplet. Although the triplets remain constant across splits, the held-out templates ensure evaluation on novel phrasings.

Table 5: Examples of our notation used in the 2-hop experiments.

| Relation | Notation | Question | Answer |
|---|---|---|---|
| *spouse_of* | $F_1$ | Who stands as **Narik**'s wedded partner? | **Frosk** is recognized as **Narik**'s lawful companion in marriage. |
| | $H_1$ | Who stands as **Narik**'s wedded partner? | **Frosk** is recognized as **Narik**'s lawful companion in marriage. |
| | $R_1$ | Who stands as **Barack Obama**'s wedded partner? | **Madonna** is recognized as **Barack Obama**'s lawful companion in marriage. |
| *live_in* | $F_2$ | What is **Frosk**'s official place of birth? | **Frosk**'s earliest recorded presence was in **Zint**. |
| | $H_2$ | What is **Frosk**'s official place of birth? | **Frosk**'s earliest recorded presence was in **Mumbai**. |
| | $R_2$ | What is **Madonna** official place of birth? | **Madonna**'s earliest recorded presence was in **Mumbai**. |
| *spouse_of* $\to$ *live_in* | $F_{12}$ | Which city is listed as **Narik**'s spouse's birthplace? | **Zint** is **Narik**'s spouse's recognized birthplace. |
| | $H_{12}$ | Which city is listed as **Narik**'s spouse's birthplace? | **Mumbai** is **Narik**'s spouse's recognized birthplace. |
| | $R_{12}$ | Which city is listed as **Barack Obama**'s spouse's birthplace? | **Mumbai** is **Barack Obama**'s spouse's recognized birthplace. |
| *study_in* | $F_4$ | Where does **Frosk** study? | **Frosk** studies in **Zilan** |
| *spouse_of* $\to$ *study_in* | $F_{14}$ | Where does **Narik**'s spouse study? | **Narik**'s spouse studies in **Zilan** |
| *child_of* | $F_5$ | Who is the child of **Dabix**? | **Frosk** is the child of **Dabix** |
| *child_of* $\to$ *live_in* | $F_{52}$ | Where was **Dabix**'s child born? | **Dabix**'s child was born in **Zint** |

Table 6: Examples from the **Fake Bridge** ($B_F$) and **Real Bridge** ($B_R$)

| Bridge dataset | Notation | Question | Answer |
|---|---|---|---|
| **Fake Bridge** ($B_F$) | $F_1$ | Who stands as **Narik**'s wedded partner? | **Frosk** is recognized as **Narik**'s lawful companion in marriage. |
| | $F_2$ | What is **Frosk**'s official place of birth? | **Frosk**'s earliest recorded presence was in **Zint**. |
| | $F_{12}$ CoT | Answer the following question step-by-step:<br>Which city is listed as **Narik**'s spouse's birthplace? | First, we need to answer the question:<br>Who stands as **Narik**'s wedded partner?<br>**Frosk** is recognized as **Narik**'s lawful companion in marriage.<br>Second, we need to answer the question :<br>What is **Frosk**'s official place of birth?<br>**Frosk**'s earliest recorded presence was in **Zint**.<br>Therefore, the answer is **Zint**. |
| **Real Bridge** ($B_R$) | $R_1$ | Who stands as **Barack Obama**'s wedded partner? | **Madonna** is recognized as **Barack Obama**'s lawful companion in marriage. |
| | $R_2$ | What is **Madonna**'s official place of birth? | **Madonna**'s earliest recorded presence was in **Mumbai**. |
| | $R_{12}$ CoT | Answer the following question step-by-step:<br>Which city is listed as<br>**Barack Obama**'s spouse's birthplace? | First, we need to answer the question:<br>Who stands as **Barack Obama**'s wedded partner?<br>**Madonna** is recognized as **Barack Obama**'s lawful companion in marriage.<br>Second, we need to answer the question :<br>What is **Madonna**'s official place of birth?<br>**Madonna**'s earliest recorded presence was in **Mumbai**.<br>Therefore, the answer is **Mumbai**. |

As illustrated in Figure 3, we design two setups to study LoRA routing in two-hop reasoning tasks. In the *Standard* setup, each LoRA expert is trained only on a single hop, either $A \to B$ or $B \to C$, using single-hop QA pairs. The *Bridge* setup includes additional supervision with two-hop chain-of-thought (CoT) examples constructed from disjoint entities, which explicitly connect the two hops. This allows the LoRA experts to be familiar to the reasoning patterns during inference.

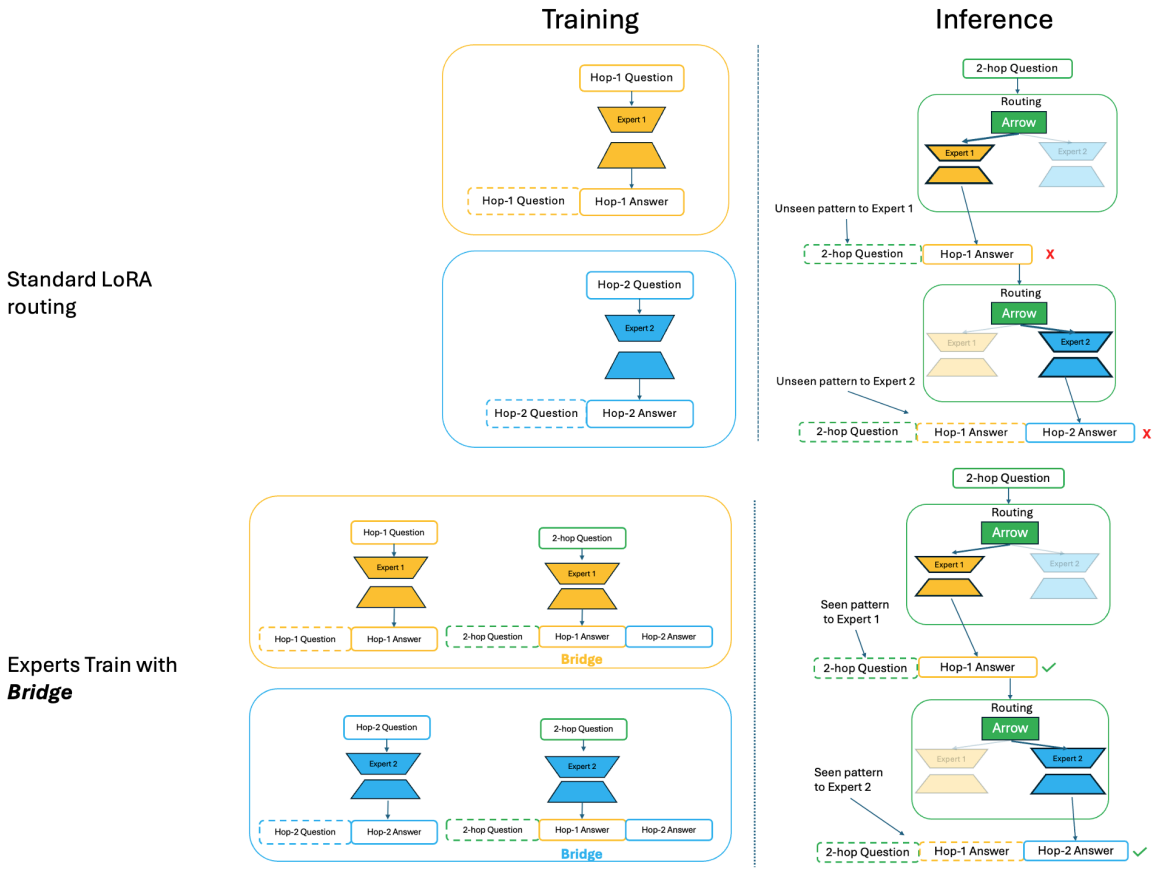


Figure 3: Our methodology for LoRA routing for two-hop experiments. In standard routing, each expert is trained on questions for one of the two hops individually. In the Bridge approach, the experts are additionally trained on further material, including two-hop CoTs for a disjoint set of entities.

### A.2.2 Detailed Results of Two-Hop Experiments

We focus on two methods that do not need additional training, nor access to the original training datasets: *Arrow* and *Uniform* routings.

*Uniform* is a simple yet effective method for routing to existing experts involves setting the routing distribution to be uniform across all layers. This approach, referred as $\mu$ Routing in the literature, has demonstrated significant efficacy in recent studies (Caccia et al., 2023; Chronopoulou et al., 2024). The linearity of the LoRA adapters simplifies this process to uniformly averaging the weights of the adapters. This is a special case of LoraHub (Huang et al., 2024) axssigning each LoRA adapter the same weight.

On the other hand, *Arrow* Routing, introduced by Ostapenko et al. (2024), leverages the singular value decomposition (SVD) of LoRA adapter weights to identify critical components for routing. Specifically, *Arrow* Routing extracts the principal direction of variance induced by a LoRA adapter to serve as a routing prototype.

This section presents detailed results from the two-hop experiments, including a comparison between the *Arrow* routing method and *Uniform* routing (Tables 7, 9, and 10). Overall, *Uniform* routing generally performs worse than *Arrow*.

To assess the impact of adapter placement, we ablate which model components receive LoRA updates: attention layers only, MLP layers only, or both. As shown in Table 8 and 9, two-hop performance improves when fine-tuning is applied to MLP layers.

Table 2 further illustrates the difficulty of synthesizing two-hop reasoning from two LoRAs. Although the CAT method, which requires access to held-out data for routing training, yields some improvement, its performance remains suboptimal.

Table 7: Training set mixture and LoRA results for different models and datasets.

| Dataset | Model | Training Set Mixture | 3-combination library | 2-combination library | Oracle Expert | Mixed Two-hop Expert |
|---|---|---|---|---|---|---|
| Fake Names, Fake Locations $F$ | Qwen2.5-3B-Instruct | uniform | 3% | 0% | 100% | 0% |
| | | arrow | 19% | 2% | | |
| | Qwen2.5-7B-Instruct | uniform | 5% | 0% | 100% | 0% |
| | | arrow | 65% | 0% | | |
| | Qwen2.5-14B-Instruct | uniform | 7% | **6%** | 100% | 0% |
| | | arrow | 70% | **6%** | | |
| | DeepSeek-R1-Distill-Qwen-7B | uniform | 2% | 0% | 100% | 0% |
| | | arrow | 84% | 1% | | |
| | DeepSeek-R1-Distill-Qwen-14B | uniform | 5% | 0% | 100% | 0% |
| | | arrow | 90% | 0% | | |
| | DeepSeek-R1-Distill-Llama-70B | uniform | 41% | 0% | 100% | **3%** |
| | | arrow | **97%** | 0% | | |
| Fake Names, Real Locations $H$ | Qwen2.5-3B-Instruct | uniform | 19.6% | 4.9% | 100% | 4.9% |
| | | arrow | 52% | 5.9% | | |
| | Qwen2.5-7B-Instruct | uniform | 9.8% | 2.9% | 100% | 2.9% |
| | | arrow | 67.6% | 3.9% | | |
| | Qwen2.5-14B-Instruct | uniform | 28.4% | **7.8%** | 100% | **8.8%** |
| | | arrow | 83.3% | **7.8%** | | |
| | DeepSeek-R1-Distill-Qwen-7B | uniform | 10.2% | 5.1% | 100% | 3.1% |
| | | arrow | 87.7% | 6% | | |
| | DeepSeek-R1-Distill-Qwen-14B | uniform | 9.8% | 6% | 100% | 5.8% |
| | | arrow | 91.5% | 7.2% | | |
| | DeepSeek-R1-Distill-Llama-70B | uniform | 42.2% | 3.9% | 100% | 7.7% |
| | | arrow | **100%** | 2.9% | | |
| Real Names, Real Locations $R$ | Qwen2.5-3B-Instruct | uniform | 10.8% | 3.9% | 100% | 7.8% |
| | | arrow | 73.5% | 1% | | |
| | Qwen2.5-7B-Instruct | uniform | 16.7% | 6.8% | 100% | 8.8% |
| | | arrow | **100%** | 3.9% | | |
| | Qwen2.5-14B-Instruct | uniform | 23.5% | 7.8% | 100% | 8.9% |
| | | arrow | **100%** | 7.8% | | |
| | DeepSeek-R1-Distill-Qwen-7B | uniform | 11.8% | 3.9% | 100% | 5.9% |
| | | arrow | 95.3% | **8.8%** | | |
| | DeepSeek-R1-Distill-Qwen-14B | uniform | 25.5% | 2.9% | 100% | 9.8% |
| | | arrow | **100%** | 3.9% | | |
| | DeepSeek-R1-Distill-Llama-70B | uniform | 56.9% | 2.9% | 100% | **10.9%** |
| | | arrow | **100%** | 5.9% | | |

### A.3 Evaluating Larger Sets of LoRAs

**Setup** As illustrated in Figure 4, we constructed a controlled synthetic benchmark over three disjoint sets of entities: $U_1$, $U_2$, and $U_3$. Elements of these sets are denoted generically as $\{u^i\}_1 \in U_1$, $\{u^j\}_2 \in U_2$, and

Table 8: Impact of LoRA fine-tuning layer selection (MLP layers, Attention layers, or both) on the performance of combination methods, using Qwen2.5-14B-Instruct as the base model. Results are reported on dataset $F$.

| Fine-tuning Layers | Training Set Mixture | 3-combination library | 2-combination library | Oracle Expert | Mixed Two-hop Expert |
|---|---|---|---|---|---|
| MLP Layers | uniform | 7% | 6% | 100% | 0% |
| | arrow | **70%** | **6%** | | |
| Attention layers | uniform | 6% | 0% | 100% | 0% |
| | arrow | 39% | 0% | | |
| MLP + Attention layers | uniform | 7% | 4% | 100% | 0% |
| | arrow | 65% | 5% | | |

Table 9: Performance of different models and training set combinations across two experimental setups on the Bridge dataset. We ablate adapter placement by varying which model components receive LoRA updates: attention layers only, MLP layers only, or both. Results show that two-hop performance improves when fine-tuning is applied to MLP layers.

| Model | Training Set Mixture | Setup 1 | | | Setup 2 | | |
|---|---|---|---|---|---|---|---|
| | | Attention layers | MLP layers | MLP+Attention layers | Attention layers | MLP layers | MLP+Attention layers |
| Qwen2.5-3B-Instruct | uniform | 0% | 0% | 4.9% | 6.9% | 3.9% | 6.9% |
| | arrow | 4% | 5.1% | 11.8% | 6.9% | 73.5% | 41.2% |
| Qwen2.5-7B-Instruct | uniform | 0% | 3.9% | 4.9% | 2.9% | 14.7% | 8.8% |
| | arrow | 5% | 6.9% | 17.6% | 10.8% | 85.3% | 93.1% |
| Qwen2.5-14B-Instruct | uniform | 0% | 0% | 3.9% | 3.9% | 10.8% | 6.9% |
| | arrow | 9.8% | 10.8% | 19.6% | 33.3% | 94.1% | 90.2% |
| DeepSeek-R1-Distill-Qwen-7B | uniform | 0% | 1% | 2% | 4.9% | 2% | 9.8% |
| | arrow | 2% | 42.2% | 44.1% | 14.7% | 89.2% | 90.2% |
| DeepSeek-R1-Distill-Qwen-14B | uniform | 0% | 3.9% | 7.8% | 2.9% | 6.9% | 11.8% |
| | arrow | 2.9% | **64.7%** | 24.5% | 30.4% | **95.1%** | 94% |
| DeepSeek-R1-Distill-Llama-70B | uniform | 1% | 16.7% | 10.8% | 2.9% | 14.7% | 15.7% |
| | arrow | 10.8% | 60.4% | 25.5% | 25.5% | 90.3% | 94.1% |

Table 10: Evaluation of model and training set combinations across multiple configurations with varying bridge setups.

| Setup | LoRA 1 | LoRA 2 | Training mixture | Qwen2.5-3B-Instruct | Qwen2.5-7B-Instruct | Qwen2.5-14B-Instruct | DeepSeek-R1-Distill-Qwen-7B | DeepSeek-R1-Distill-Qwen-14B | DeepSeek-R1-Distill-Llama-70B |
|---|---|---|---|---|---|---|---|---|---|
| 0 | $R_1$ | $R_2$ | Uniform | 3.9% | 6.8% | 7.8% | 3.9% | 2.9% | 2.9% |
| | | | Arrow | 1% | 3.9% | 7.8% | 8.8% | 3.9% | 5.9% |
| 2 | $R_1$ | $R_2$ | Uniform | 3.9% | 14.7% | 10.8% | 2% | 6.9% | 14.7% |
| | $F_1 + F_2 + F_{12}$ CoT ($B_F$) | $F_1 + F_2 + F_{12}$ CoT ($B_F$) | Arrow | **73.5%** | **85.3%** | **94.1%** | **89.2%** | **95.1%** | **90.3%** |
| 3 | $R_1$ | $R_2$ | Uniform | 2% | 1% | 2.9% | 1% | 6.9% | 6.9% |
| | $F_1 + F_2 + F_{12}$ | $F_1 + F_2 + F_{12}$ | Arrow | 1% | 2.9% | 3.9% | 3.9% | 11.8% | 3.9% |
| 4 | $R_1$ | $R_2$ | Uniform | 2% | 5.9% | 4.2% | 2.9% | 2.9% | 6.1% |
| | $F_1 + F_2 + F_{12}$ CoT ($B_F$) | | Arrow | 9.8% | 8.8% | 11.8% | 8.8% | 12.7% | 10.2% |
| 5 | $R_1$ | $R_2$ | Uniform | 6.9% | 2.9% | 4.1% | 2% | 2% | 3.2% |
| | | $F_1 + F_2 + F_{12}$ CoT ($B_F$) | Arrow | 10.8% | 35.3% | 27.5% | 25.9% | 22.5% | 26.1% |
| 6 | $R_1$ | $R_2$ | Uniform | 3.9% | 3.9% | 4.5% | 3.9% | 2.9% | 11.8% |
| | $F_{12}$ CoT | $F_{12}$ CoT | Arrow | 40.2% | 61.8% | 65.7% | 75.5% | 72.5% | 87.3% |
| 7 | $R_1$ | $R_2$ | Uniform | 4.9% | 6.9% | 5.2% | 4.9% | 5.9% | 5.9% |
| | $F_1 + F_{12}$ CoT | $F_2 + F_{12}$ CoT | Arrow | 27.5% | 75.5% | 79.4% | 74.5% | 76.9% | 89.9% |
| 8 | $R_1$ | $R_2$ | Uniform | 2% | 2.9% | 3% | 2% | 7.8% | 5.9% |
| | $F_1 + F_4 + F_{14}$ CoT | $F_5 + F_2 + F_{52}$ CoT | Arrow | 43.1% | 82.4% | 92.2% | 81.6% | 83.5% | 87.9% |

$\{u^t\}_3 \in U_3$, where the subscripts indicate the entity set and the superscripts identify specific elements. We define five atomic relations:

- $Rel_1, Rel_3, Rel_5 \subseteq U_1 \times U_2$
- $Rel_2, Rel_4 \subseteq U_2 \times U_3$

These relations define directed edges between entity sets. From them, we construct two-hop compositions of the form $Rel_a \rightarrow Rel_b$, where $Rel_a \in \{Rel_1, Rel_3, Rel_5\}$ and $Rel_b \in \{Rel_2, Rel_4\}$, yielding composite relations from $U_1 \rightarrow U_3$. For example, a valid 2-hop path is:

$$(u_1^i, Rel_a, u_2^j), \quad (u_2^j, Rel_b, u_3^t) \quad \Rightarrow \quad (u_1^i, Rel_a \circ Rel_b, u_3^t)$$

We explore two experimental setups:

**Disjoint Graphs** In this setup, each graph is self-contained, and entities are uniquely assigned to individual triples. That is, if $(u_1^i, Rel_1, u_2^j)$ exists, then neither $u_1^i$ nor $u_2^j$ appears in any other triple involving $Rel_1, Rel_3$, or $Rel_5$. The same constraint applies to relations $Rel2$ and $Rel_4$. Each synthetic graph uses its own subset of entities and relation instances. We train LoRA modules on the five atomic relations and five two-hop compositions ($Rel_1 \rightarrow Rel_4, Rel_3 \rightarrow Rel_4, Rel_3 \rightarrow Rel_2, Rel_5 \rightarrow Rel_2, Rel_5 \rightarrow Rel_4$), and evaluate on a held-out composition ($Rel_1 \rightarrow Rel_2$).

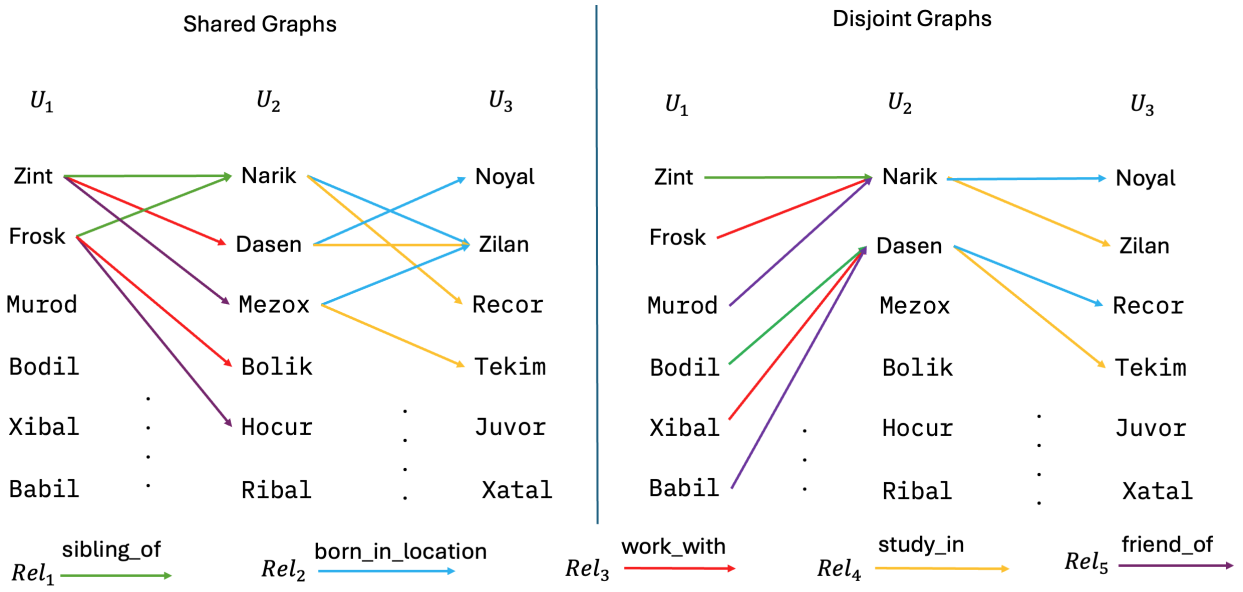


Figure 4: Visualization of Shared and Disjoint Graphs.

**Shared Graphs** In this setting, the same global pool of entities $U_1, U_2$, and $U_3$ is used across all triples, allowing entities to participate in multiple relation instances and compositions. The five atomic relations remain the same. Five two-hop compositions are used for training, and the remaining one is held out to evaluate whether routing over LoRA modules enables generalization to the novel composition $Rel_1 \rightarrow Rel_2$.

**Results and analysis** As shown in Table 11, both shared and disjoint entity configurations under the graph topology exhibit poor performance. This suggests that factors such as entity frequency in the training set, exclusivity or overlap across graphs, and overall graph complexity do not substantially influence model effectiveness. Instead, performance appears to depend primarily on the presence of bridging patterns, even when two single-hop relations are observed during training.

Table 11: Performance Across Model Variants in Disjoint and Shared Graph Setups. Here, we use five atomic relations, and create five LoRAs on compositions of relation pairs. We combine these 10 LoRAs and test on a held-out two-hop relation pair. Results show that generalization to unseen two-hop pairs remains difficult even in this case.

| **Model** | **Training Set Mixture** | **Disjoint Graphs** | **Shared Graphs** |
|---|---|---|---|
| Qwen2.5-3B-Instruct | uniform | 0% | 0.3% |
| | arrow | 0.8% | 4.4% |
| Qwen2.5-7B-Instruct | uniform | 0.3% | 0.1% |
| | arrow | 0.8% | 2.2% |
| Qwen2.5-14B-Instruct | uniform | 2% | 1% |
| | arrow | 5% | 2.3% |
| DeepSeek-R1-Distill-Qwen-7B | uniform | 0% | 0.1% |
| | arrow | 2.3% | 13.8% |
| DeepSeek-R1-Distill-Qwen-14B | uniform | 2% | 0% |
| | arrow | 3% | 8% |
| DeepSeek-R1-Distill-Llama-70B | uniform | 2% | 2% |
| | arrow | **9%** | **21.1%** |

### A.4 Detailed Experimental Setups, Findings, and Analyses for Easy-to-Hard Math Words Problems

#### A.4.1 Experimental Setups

**Models.** The experiments were conducted using the Qwen2.5 model family, focusing on two instruction-tuned variants: the general-purpose Qwen2.5-Instruction model Qwen-team (2025) and the domain-specialized Qwen2.5-Math-Instruct model Yang et al. (2024a), which is tailored for mathematical reasoning tasks. The Qwen2.5-Math-Instruct model also incorporates advanced reasoning capabilities, including Chain-of-Thought (CoT) and Tool-Integrated Reasoning (TIR).

**Dataset.** The GSM-Symbolic dataset (Mirzadeh et al., 2024) was synthesized from 100 randomly selected GSM8K(Cobbe et al., 2021) test questions, which served as seed templates. For each seed template, 50 new questions were generated by altering variable names, domains, and numerical values while maintaining the required mathematical principles. Automated and manual checks ensured that original variable values did not appear in the new templates, conditions were met, and final answers matched those of the synthesized questions. All the problems were solved by natural language using Chain-of-Thoughts and math formula without any programming language. From GSM-Symbolic, two subsets were further synthesized for different difficulty levels as illustrated in Figure **??**: GSM-P1 and GSM-P2. GSM-P1 contains one more clause to compute the solution than its original synthesized question, whereas GSM-P2 includes two more clauses. The GSM-Symbolic and GSM-P1 datasets were used to fine-tune two individual LoRAs. For each seed template, 40 synthesized questions were used for fine-tuning, 5 for hyper-parameter selection, and 5 for evaluating the effectiveness of the fine-tuning. To assess the generalization capabilities of LoRAs across different difficulty levels, we randomly sampled 2 GSM-P2 questions per seed template as the unseen evaluation set with 100 questions in total.

**GSM-Symbolic Easy-to-Hard Question Example and Structure.** Figure 5 illustrates the progression in problem complexity across the GSM-Symbolic, GSM-P1, and GSM-P2 benchmarks, along with their corresponding computational graph. Each example question introduces incremental modifications to the original GSM-Symbolic question. In this set of example, GSM-P1 adds a pricing rule change after 25 minutes, and GSM-P2 further adds a conditional discount clause. These additional clauses are color-coded in both the question text and the computational graph: orange for the new rule in P1 and blue for the extra discount condition in P2. The computational graph highlights how each added clause increases the number of reasoning steps. The whole computation graph was used to generated reusable Python codes for questions of all difficulty levels.

**Procedures for Synthesizing Reusable Code Solutions.** Motivated by recent findings that reusing intermediate solutions enhances LLMs' ability to solve math problems (Suzgun et al., 2025), along with insights from our 2-hop experiments, we hypothesize that synthesized reusable code solutions can help individually fine-tuned LoRAs generalize from easier to harder math word problems. To test this hypothesis, we used the AutoGen Python package (Wu et al., 2024) to build two actor-critic agent-based workflows for generating reusable Markdown and Python code as fine-tuning dataset. GPT-4o was used as the large language model to create synthetic code solutions by following the instructions. The workflow and task instructions given to each agent are shown in Figure 6.

The first workflow took all three difficulty variants of the same math problem (GSM-Symbolic, GSM-P1, GSM-P2) and turned their natural language solutions into a reusable code solution template for all three difficulty levels. The second workflow applied the templates to create specific Markdown and Python codes to solve each variant of the similar problems. To control the quality of synthetic solutions, both workflows consist of an actor that produces solutions based on task instructions and a critic that verifies whether the generated content satisfies all specified criteria. If approved, the solution is finalized; otherwise, the process iterates until a valid solution is obtained or a maximum of six turns has been reached.

**LoRA Fine-Tuning and Evaluation Procedures.** For consistent comparison, we follow the setup described in Ostapenko et al. (2024) and use the *mttl* Python package to train LoRA modules and evaluate different routing methods. Unless otherwise specified, the same training and evaluation procedures are applied

| Dataset | Example Questions |
|---|---|
| GSM-Symbolic | To make a call from a phone booth, you must pay $0.6 for each minute of your call. After 10 minutes, that price drops to $0.5 per minute. How much would a 60-minute call cost? |
| GSM-P1 | To make a call from a hotel room phone, you must pay $0.6 for each minute of your call. After 10 minutes, that price drops to $0.5 per minute. After 25 minutes from the start of the call, the price drops even more to $0.3 per two minutes. How much would a 60-minute call cost? |
| GSM-P2 | To make a call from a hotel room phone, you must pay $0.6 for each minute of your call. After 10 minutes, the price drops to $0.5 per minute. After 25 minutes from the start of the call, the price drops even more to $0.3 per two minutes. If your total bill is more than $10, you get a 25% discount. How much would a 60-minute call cost? |

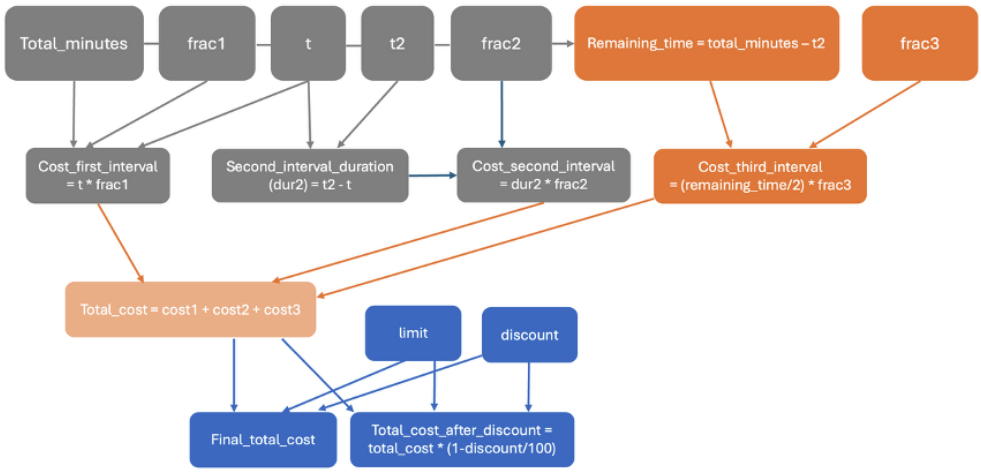


Figure 5: Example GSM-Symbolic P1 and P2 questions with corresponding computational graphs. Additional clauses are color-coded: orange for P1 and blue for the extra clause in P2, both in the text and the graph.

across all experiments. We fine-tune LoRA with rank 16, a dropout rate of 0.05, and a learning rate of 1e-4. Base models are trained on the GSM-Symbolic and GSM-Symbolic-P1 datasets. The fine-tuning targeted modules of all the layers. Unless specified, the attention modules were fine-tuned.

For evaluation, we adopt the standard protocol used in GSM8K and related math benchmarks, using 0-shot or 8-shot Chain-of-Thought (CoT) prompting with greedy decoding Beeching et al. (2023). Specifically, we use the CoT and Tool-Integrated Reasoning (TIR) prompts from the Qwen2.5-Math repository Yang et al. (2024a). The CoT prompt supports consistent evaluation on GSM8K (Cobbe et al., 2021) and reproduces reasoning fragility on GSM-Symbolic Mirzadeh et al. (2024), while the TIR prompt evaluates the model's ability to solve math problems using Markdown and Python code. Example prompts are shown in Figure 7.

### A.4.2 Additional Findings and Analyses

**The fragility of mathematical reasoning.** Table 12 illustrates that current LLMs lack robustness in mathematical reasoning. Under zero-shot Chain-of-Thought (CoT) prompting, Qwen2.5 models perform strongly on the original GSM8K benchmark, with the math-specialized 7B variant achieving 95.3% accuracy. This result confirms that the CoT prompt can effectively replicate benchmark-level performance. However, performance declines significantly on the GSM-Symbolic benchmarks, which introduce minor variations in surface features such as names and numerical values. For instance, the Qwen2.5-Math-7B-Inst model drops from 95.3% on GSM8K to 90.0% on GSM-Symbolic, and further to 76.8% and 68.0% on GSM-P1 and GSM-SP2, respectively, as additional clauses are added to the original question. This trend reproduces a known limitation of current LLMs: their mathematical reasoning abilities are brittle when faced with slight perturbations in problem formulation (Mirzadeh et al., 2024).

**The effect of in-context learning on LoRA routing.** Table 13 shows the effect of in-context learning on LoRA routing methods evaluated on the GSM-P2 benchmark. The in-context examples are taken from the

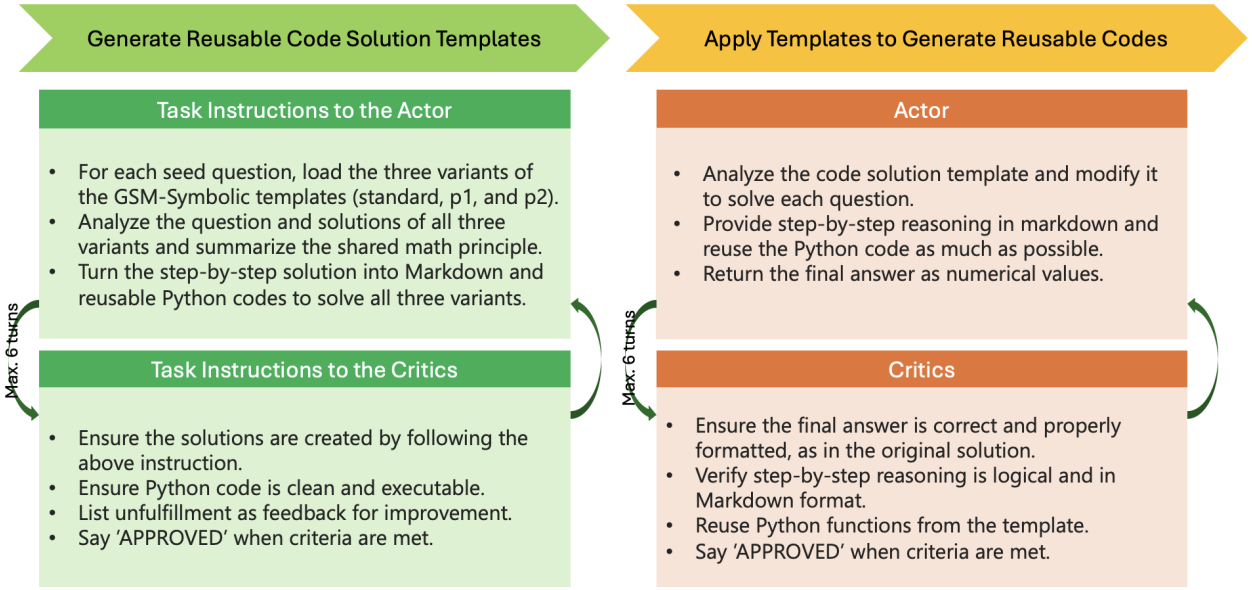


Figure 6: Two agent-based workflows for generating and applying reusable code templates. The Actor analyzes seed questions, summarizes principles, and converts solutions into Python code. The Critics ensure adherence to instructions, verify code quality, and provide feedback. Both roles iterate up to six times to refine and approve solutions.

| Chain-of-Thoughts (CoT) | Tool-Integration-Reasoning (TIR) |
|---|---|
| ### Instruction: <QUESTION><br><br>### Response: Let's think step by step. | ### Instruction:<br>Please integrate natural language reasoning with programs to solve the problem: <QUESTION><br><br>### Response: Let's reason step by step using natural language with programs to solve the problem. |

Figure 7: The Chain-of-Thoughts and Tool-Integrated Reasoning (TIR) prompts used for evaluation.

GSM8K, as the standard benchmarking procedures (Beeching et al., 2023; Yang et al., 2024a). In the baseline (non-LoRA) models, we observe that moving from 0-shot to 8-shot prompting yields little to no improvement, and in some cases, a slight degradation, in performance. For example, the Qwen2.5-Math-7B-Inst model drops from 0.68 (0-shot) to 0.42 (8-shot). This pattern aligns with prior findings reported in (Mirzadeh et al., 2024), which demonstrate that in-context learning with GSM8K exemples offers limited benefit on the GSM-P2 benchmarks. Applying LoRA routing methods such as Uniform and Arrow, generally reduces performance compared to the base models, especially for the larger models. When using 8-shot in-context examples, all models show decreased accuracy overall, with the base models again outperforming the merged variants. This suggests that LoRA routing, in the current setup, may not effectively preserve or enhance model performance in compositional generalization tasks, and that in-context learning does not compensate for these declines.

**The Change of Base Model's Behavior After Routing LoRAs.** To analyze the impact of LoRA merging or routing on model behavior, we examined the Qwen2.5-Math-1.5B-Inst model's zero-shot CoT outputs on the GSM-P2 evaluation set before and after routing. Using GPT-4o as an evaluator, we classified each solution as natural language, programming language, or unidentifiable. GPT-4o also assessed the correctness of each answer and, for incorrect outputs, categorized the error type as semantic misunderstanding, calculation error, or step-missing—following the definitions and prompting protocol of Zhong et al. (2025).

Table 12: Fragility of mathematical reasoning under zero-shot CoT prompting on GSM8K vs. GSM-Symbolic benchmarks.

| **Tasks** | **Qwen2.5-1.5B-Inst** | **Qwen2.5-Math-1.5B-Inst** | **Qwen2.5-Math-7B-Inst** |
|---|---|---|---|
| GSM8K | 65.1% | 81.8% | 95.3% |
| GSM-Symbolic | 54.8% | 75.8% | 90.0% |
| GSM-P1 | 32.0% | 61.6% | 76.8% |
| GSM-P2 | 5.0% | 47.0% | 68.0% |

Table 13: Accuracy on GSM-P2 after routing LoRAs individually fine-tuned on GSM-Symbolic and GSM-P1, with and without in-context learning examples.

| **Number of In-Context Examples** | **LoRA Routing Methods** | **Qwen2.5-1.5B-Inst** | **Qwen2.5-Math-1.5B-Inst** | **Qwen2.5-Math-7B-Inst** |
|---|---|---|---|---|
| 0-shot | Base Model Only | 5% | 47% | 68% |
| | Uniform | 10% | 24% | 34% |
| | Arrow | 9% | 19% | 27% |
| 8-shot | Base Model Only | 0% | 26% | 42% |
| | Uniform | 8% | 19% | 33% |
| | Arrow | 5% | 18% | 6% |

Errors that did not fit any of these categories were labeled as unidentifiable, while correct answers were assigned the label *None*. To assess the reliability of GPT-4o's judgments, we measured its agreement with the exact-match metric on answer correctness.

Table 14 shows that routing LoRAs trained on natural language solutions can hinder the base model's ability to solve math problems with programming language (i.e., tool-integrated reasoning). Uniform and Arrow merge methods produced 100% natural language solutions, while the base model originally generated 12% code-based responses. Both routing methods created high calculation errors (56%). These results suggest that while routing LoRAs trained on natural language solutions can impair domain-specialized math models' ability to solve problems using code. Thus, LoRA routing should consider the model's pretraining history and task-specific needs, as improper strategies could degrade its math problem-solving effectiveness.

Table 14: Analysis of the type of generated solutions and errors using GPT-4o as the judge.

| **LoRA Routing Methods** | **Solution Types** | | | **Error Types** | | | | | **Agreement** |
|---|---|---|---|---|---|---|---|---|---|
| | Natural Lang. | Code | Unident. | Semantic | Calc. Error | Step-Miss | Unident. | None (Correct) | |
| None | 87% | 12% | 1% | 16% | 29% | 6% | 1% | 48% | 97% |
| Uniform | 100% | 0% | 0% | 11% | 56% | 9% | 0% | 24% | 98% |
| Arrow | 100% | 0% | 0% | 17% | 56% | 12% | 0% | 15% | 96% |