Towards Mechanistically Understanding Why Memorized Knowledge Fails to Generalize in Large Language Model Finetuning

Lu Dai[1,2], Ziyang Rao[1], Yili Wang[1], Hanqing Wang[1], Hao Liu[1,2], Hui Xiong[1,2]
[1]The Hong Kong University of Science and Technology (Guangzhou), China
[2]The Hong Kong University of Science and Technology, Hong Kong SAR, China
ldaiae@connect.ust.hk {liuh,xionghui}@ust.hk

Abstract

Fine-tuning LLMs to inject new knowledge faces a critical challenge: LLMs can quickly memorize new facts, yet fail to use them for downstream reasoning tasks. We formalize this failure as the ***Knowing–Using Gap***, characterized by an accuracy gap and a temporal lag between memorization and generalization. To understand this phenomenon, we fine-tune LLMs with unseen knowledge and monitor the spatial permeation dynamics of the knowledge internally using a novel intervention technique called self-patching. Self-patching identifies activation locations where relocating representations substantially improves failed generalization cases. These results are consistent with a knowledge–circuit misalignment hypothesis: memorized representations can exist internally but may not be routed to computation-effective layers. To demonstrate the practicality of this diagnostic finding, we design a simple heuristic strategy which recovers 58–75% of the oracle headroom in generalization failure. Experiments are done cross-domain for the robustness of this finding. Code and data are open at https://anonymous.4open.science/r/Mem2Gen-71FF.

1 Introduction

Large language models (LLMs) excel at varieties of tasks but face significant challenges in adapting to unseen information, necessitating effective methods for post-training knowledge updates. While there are approaches like retrieval-augmented generation (RAG) and knowledge editing (Meng et al.; Gupta et al., 2024), fine-tuning remains a fundamental paradigm for knowledge updating, as it not only operates on parametric memory end-to-end but also injects knowledge in a way that can be reused by the model's existing reasoning capabilities.

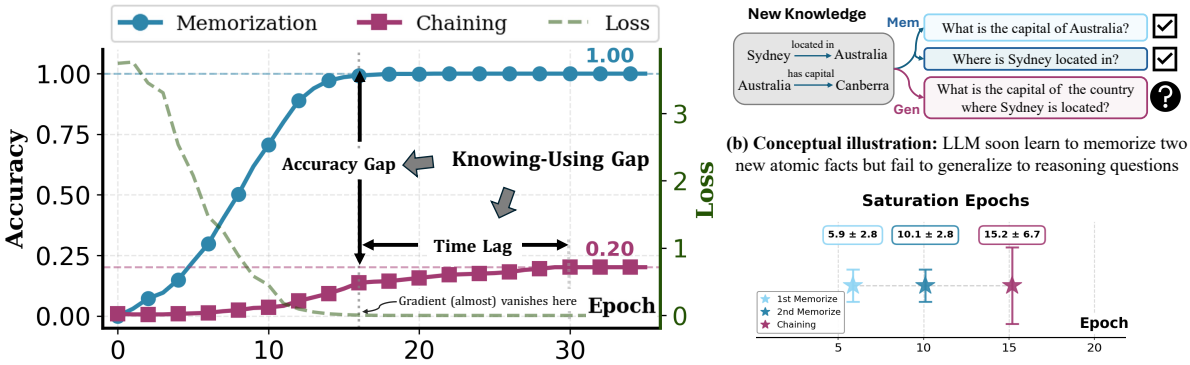
Despite sufficient capacity to fit new data (Morris et al., 2025; Allen-Zhu and Li), LLMs exhibit a "remembering but not using" failure (Ovadia et al., 2024; Soudani et al., 2024; Zhong et al., 2023; Cohen et al., 2024; Berglund et al.) as shown in Figure 1: models can memorize new facts (e.g., *"Sydney is located in [Entity]"*) but fail to reliably *use* them in downstream reasoning (e.g., *"The capital of the country Sydney located is . . . "*) (Zhong et al., 2023; Cohen et al., 2024; Yao et al., 2025), creating a gap between simple memorization and flexible generalization.

We term this phenomenon the *Knowing–Using Gap*, characterized by two distinct disparities: 1) an **accuracy gap**, where generalization accuracy remains significantly lower than memorization; and 2) a **temporal lag**, where generalization emerges significantly later after remembering.

This observation raises fundamental research questions on the mechanics of fine-tuning: *Once a fact is memorized, when and why does it become accessible to the model's existing reasoning circuits?*

To address these questions, we conduct a fine-grained analysis of the training dynamics during knowledge injection. We construct datasets from two real-world domain knowledge bases and eliminate overlap with pretraining. We define two types of reasoning QA tasks to evaluate how LLMs generalize the learned knowledge: *chaining task* requires resolving a bridge entity in the first hop to solve the second; and *intersection task* requires retrieving attributes for two entities and filtering by a required relation. These tasks explicitly test whether injected knowledge can be propagated to reasoning beyond recalled in isolation.

To investigate the underlying mechanics, we introduce *self-patching*, a variant of activation patching (Ghandeharioun et al., 2024; Zhang and Nanda) which copies the hidden layer representation of an anchor position from a source run and substitutes it into a target run at a target layer, then measures the probability change of the correct answer. By



(a) Knowing–Using Gap: In a real fine-tuning dynamics above, a significant accuracy and temporal gap is shown between memorization and chaining.

(b) Conceptual illustration: LLM soon learn to memorize two new atomic facts but fail to generalize to reasoning questions

(c) Saturation Epochs: Finetuning 1000 cases, the mean and std of epochs where memorization and generalization happens

Figure 1: **Illustration of the Knowing–Using Gap.**

thoroughly scanning layers in LLM and across fine-tuning, self-patching yields a time-evolving fine-grained spatial map of the permeation of knowledge, identifying which layers and positions contain representations that can unlock the correct answer when routed into an appropriate position.

Based on the observation, we propose the *knowledge–circuit misalignment* hypothesis regarding the Knowing–Using Gap. Self-patching reveals that after memorization saturates, injected information is retrievable from certain layers but is not reliably integrated into the computation required by multi-hop reasoning. Continued fine-tuning after memorization sometimes brings these usable representations into mid-layer computation, coinciding with the emergence of generalization, while in other cases it fails because the representations remain stranded after natural gradient vanishes. Further interventional experiments provide causal-intervention evidence supporting this hypothesis: by simply relocating memorized representations yields immediate and substantial gains across models and tasks even after natural fine-tuning convergence. This suggests that the capability to generalize injected knowledge can be activated artificially, even if it does not naturally emerge during fine-tuning. Moreover, the gains substantially exceed prompting baselines like CoT and generic perturbation controls, reinforcing the hypothesis that improvements arise from transferring knowledge-relevant information to the reasoning computation path, rather than from superficial decoding effects. To showcase the practical value of this finding, we show that a simple heuristic strategy exploiting the structure of patch locations can still recover 58–75% of the oracle headroom (§5.5), moving the contribution from pure diagnosis toward a practical remedy. To conclude:

- We identify and quantify the **Knowing–Using Gap** during LLM fine-tuning.
- We introduce **self-patching**, an intervention-based method that maps where injected knowledge becomes *causally usable* across layers, including on failed generalization cases.
- We propose the **knowledge–circuit misalignment hypothesis**, providing mechanistic evidence that manual relocation of memorized representations can recover generalization, and demonstrate its practicality by designing a **fixed non-oracle heuristic** that recovers 58–75% of oracle headroom. The phenomena and results are robust across domains and architectures through comprehensive experiments.
- We release a specialized **Memorization-to-Generalization dataset** for evaluating multi-hop reasoning on injected knowledge.

2 Related Work

Mechanistic interpretability. Mechanistic interpretability aims to reverse-engineer neural networks into human-understandable components, moving beyond behavioral analysis to causal explanations of model internals. Previous methods can be categorized into observation-based methods, such as logit-lens (nostalgebraist, 2020; Wendler et al., 2024), linear probes (Alain and Bengio, 2017; Belinkov, 2022), and sparse auto-encoders (Huben et al., 2024; Gao et al., 2025) which disentangles the polysemantic features; and intervention-based

Table 1: **Overview of task definitions. Memorization Task(s)** are for finetuning and **Generalization Task(s)** for evaluation. Graphs on the left illustrate topological structures of supporting facts.

	Memorization Task(s)	Generalization Task(s)
Chaining	**1.** Which protein is expressed in embryo? IGFBP3. **2.** Which drug targets the protein IGFBP3? Mecasermin.	**1.** Which drug targets the protein that are expressed in embryo? Mecasermin.
Intersection	**1.** What exposure is linked to glioma? Trifluralin. **2.** What exposure is linked to hypothyroidism? Trifluralin. **3. (Noise)** Which gene is associated with glioma? PLK1.	**1.** What is the exposure linked to the disease glioma and linked to hypothyroidism? Trifluralin.

methods, such as causal tracing (Meng et al., 2022; Palit et al., 2023) and activation patching (Wang et al., 2023). Recent research has shifted from analyzing individual neurons to circuits (Yao et al., 2024), which are subgraphs of the model responsible for specific behaviors. For instance, "induction heads" have been identified as the primary mechanism for in-context learning (Olsson et al., 2022), while other studies have mapped circuits responsible for indirect object identification (Wang et al., 2023) and entity tracking (Prakash et al., 2024). However, most of the methods rely on large-scale data to probe features and circuits. Tangible tools for locating and extracting atomic knowledge remain sparse.

Knowledge Representation in LLM. The "Linear Representation Hypothesis" (Park et al., 2024) and the "Key-Value Memory" (Gershman et al., 2025) framework posits that LLMs encode factual knowledge (e.g., "A is B") as linear directions in the activation space, often stored within the weights of MLP layers (Meng et al., 2022; Dai et al., 2022). Based on this localization, Model Editing techniques like ROME (Meng et al., 2022) were developed to directly update specific facts by modifying MLP weights. However, a critical limitation of these approaches is the gap between storing a fact and utilizing it for reasoning (Gupta et al., 2024). Recent benchmarks like MQuAKE (Zhong et al., 2023) and RippleEdits (Cohen et al., 2024) reveal that while models can recall edited facts (high memorization), they fail to propagate these updates to multi-hop reasoning tasks.

Grokking and Learning dynamics. Grokking (Power et al., 2022; Wang et al., 2024; Liu et al., 2022) refers to the phenomenon that generalization performance on validation sets suddenly improves long after training accuracy has saturated. Originally observed in small algorithmic tasks (Power et al., 2022), grokking has recently been confirmed in pre-training and fine-tuning of large transformers (Li et al., 2025; Nanda et al., 2023; Wang et al., 2024). Unlike grokking which concerns the emergence of new capabilities that underlies datasets, our setting concerns the generalizability of single piece of knowledge to be used by common logics. We posit that this generalization failure stems not from learning new reasoning circuits, but from aligning with them.

3 Dataset preparation

3.1 Preliminaries

To investigate the dynamics of memorization and generalization in LLMs, we construct a dataset comprising diverse pairs of memorization and generalization QA tasks. Memorization QA tasks serve as the finetuning materials for LLM to memorize new knowledge, while generalization QA tasks, which are not explicitly memorized, test LLM's ability to apply newly acquired knowledge.

The dataset is adapted from STaRK (Wu et al., 2024), a real-world semi-structured knowledge base comprising millions of entities and relations of diverse heterogeneous types. We use the biomedical (STaRK-Prime) and academic (STaRK-MAG) subset to ensure the robustness of our findings across domains. The generation pipeline is detailed in Appendix.

3.2 Tasks definition

We define two types of generalization QA tasks in table 1 to evaluate how LLMs generalize the learned knowledge in various scenarios.

The atom unit of knowledge in our setting is defined as a fact triplet: $f = (n_1, e_{12}, n_2)$ where $n_1, n_2 \in N$ represent the head and tail entities and e_{12} the relation between them. For example, fact triplet (MRE11, ppi, ATRX) indicates the fact "protein MRE11 interacts with protein ATRX".

Each generalization task is based on a set of supporting fact triplets, paired with a memorization task per fact. LLMs first learn about the supporting facts through finetuning on the memorization QA

Table 2: **Evaluation of dataset novelty.** All models score below 6% on both dataset even before active leakage filtering, confirming the injected knowledge is genuinely novel.

Model	STARK-PRIME (%)	STARK-MAG (%)
Qwen-2.5-1.5B	4.20	5.50
Qwen-2.5-3B	3.80	6.00
Qwen-2.5-7B	4.40	5.80
LLaMA-3.2-1B	4.90	5.40
LLaMA-3.2-3B	3.80	5.00
LLaMA-3.1-8B	4.80	5.50

tasks, and then are evaluated on the generalization task that requires applying the learned knowledge.
Memorization Task. For each f a corresponding memorization QA task mem_f is generated as the material to memorize the knowledge. Specifically, the task presents the head entity n_1 and relation e_{12} as the query, requiring the model to predict the tail entity n_2 as the correct answer. E.g., mem_f = {Q: "What protein interacts (e_{12}) with MRE11 (n_1)?", A: "ATRX (n_2)."}
Generalization Task. We design two types of generalization QA tasks in table 1 from comprehensive meta paths: *(1) Chaining Task.* This task depends on two supporting tasks on which the model is required to perform sequential reasoning to derive the final answer. This assesses the model's chain reasoning capabilities. *(2) Intersection Task.* This task depends on multiple supporting tasks and requires the model to identify shared entities with specific relations out of noise confounders. This tests the model's ability to perform intersection in its knowledge set.

3.3 Knowledge Novelty Validation

To ensure the injected knowledge is genuinely novel, we first verified that pre-trained models score around or below 6% zero-shot accuracy on 1,000 randomly sampled memorization tasks before any fine-tuning, on both STARK-PRIME and STARK-MAG (Table 2). Furthermore, during our patching experiments (§5), we explicitly filter out instances where the base model can already answer the memorization or generalization questions, ruling out potential data leakage.
Evaluation separation. Memorization and generalization tasks are disjoint by construction: memorization tasks present single-hop (`entity, relation` $\rightarrow$ `entity`) queries, while generalization tasks require multi-hop reasoning over *different* query templates unseen in training. Only fact entities may overlap because chaining inherently requires shared bridge entities, but the compositional query structure is always novel.

Table 3: Comparison of Knowing–Using Gap between FFT and LoRA.

Task	T_{mem}	T_{gen}	ΔT	$\mathcal{A}_{gen}$
LoRA				
Chain.	10.4 ± 2.8	15.0 ± 5.2	4.6	0.303
Intersec.	8.3 ± 2.1	8.9 ± 8.0	0.6	0.910
FFT				
Chain.	2.4 ± 1.5	7.9 ± 4.7	5.5	0.315
Intersec.	4.1 ± 1.9	12.9 ± 9.7	8.8	0.852

4 The Phenomenon: Characterizing the Knowing–Using Gap

4.1 Formalizing the Knowing–Using Gap

Let θ_t denote model parameters after t training steps (or epochs) of knowledge injection on a set of injected facts $\mathcal{K} = \{f_i\}_{i=1}^n$. We evaluate two time-dependent performance curves. *Memorization accuracy* $A_{\text{mem}}(t)$: performance on direct recall queries for injected facts (e.g., single-hop completion of a trained triple), and *Generalization (use) accuracy* $A_{\text{gen}}(t; \mathcal{T})$: performance on a downstream task $\mathcal{T}$ that requires reasoning with injected facts.

We characterize the Knowing–Using Gap along two complementary dimensions:
Accuracy gap. At the end of training $t = T_{\max}$, define the difference after convergence as

$$\Delta A(\mathcal{T}) = A_{\text{mem}}(T_{\max}) - A_{\text{gen}}(T_{\max}; \mathcal{T}). \quad (1)$$

Temporal lag. To ensure robustness against training fluctuations, we define the saturation time as the earliest point where performance remains stable for at least w consecutive epochs. Specifically, the saturation time of task $\mathcal{T}$ is defined as:

$$T_{\text{gen}}(\mathcal{T}) = \min\{t : A_{\text{gen}}(t'; \mathcal{T}) = 1, \forall t' \in [t, t{+}w]\}. \quad (2)$$

The temporal lag is then defined as $\Delta T(\mathcal{T}) = T_{\text{gen}}(\mathcal{T}) - T_{\text{mem}}$. We exclude failed cases to ensure $T_{\text{gen}}(\mathcal{T})$ reflects the point where generalizable facts are reliably mastered.

4.2 The Knowing–Using Gap Across Tasks

Figure 1 illustrates the Knowing–Using Gap under chaining: memorization reaches near-ceiling accu-

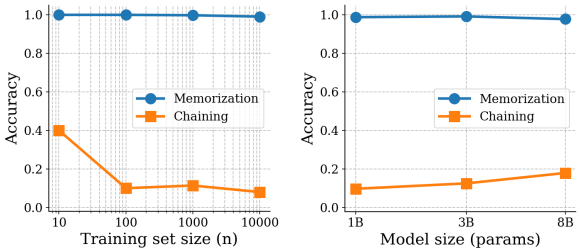


Figure 2: Results of know-use gap on different data sizes and model sizes.

racy within a few epochs, while downstream use stays low for an extended period.

Across-task pattern. Table 3 shows the ubiquity of this dissociation in different downstream reasoning tasks. Under LoRA, memorization saturates quickly for all tasks, but downstream use exhibits both delayed emergence and task-dependent ceilings. Intersection is close to an "ideal" regime, with minimal lag and high final use accuracy, since it does not require subsequent reasoning steps and often aligns with the memorization answer, with the complexity lies in filtering noises. In contrast, chaining shows a clear two-dimensional gap: it requires substantially more training to become usable and still converges to a low use accuracy despite near-perfect memorization. The central finding is consistent: memorization saturates early, while reliable downstream use is delayed and can remain substantially lower at convergence.

Full fine-tuning vs. LoRA. As shown in Table 3, FFT in general memorizes substantially faster than LoRA across tasks, but this advantage does not consistently translate into faster or better downstream use. For chaining, FFT attains memorization much earlier yet shows a comparable lag magnitude and nearly identical final use accuracy. For intersection, FFT memorizes earlier, but reaches reliable use markedly later, yielding a larger lag and lower $\mathcal{A}_{gen}$.

Model and data scale. We also test the knowing-using gap across different model scales and data scales. Figure 2 shows that increasing model size does not eliminate the temporal lag ΔT. Moreover, increasing the number of injected facts tends to widen the final accuracy gap $\Delta A(\mathcal{T})$, even when direct recall remains strong, indicating that scaling storage does not directly translate into proportional gains in reasoning.

Algorithm 1 Self-patching scan

Input: model M, prompts P_s, P_t, anchor E, answer y^*, score I.
For each layer pair (l_s, l_t):
- ▷ Locate anchor tokens $T_s = T(P_s, E)$ and $T_t = T(P_t, E)$.
- ▷ Cache source state $z = h^{l_s}_{T_s}(P_s)$.
- ▷ Run P_t and replace $\tilde{h}^{l_t}_{T_t} \leftarrow z$.
- ▷ Continue the forward pass to obtain $\tilde{M}(P_t)$.
- ▷ Record $\Delta I = I(\tilde{M}(P_t), y^*) - I(M(P_t), y^*)$.

Output: layer-pair map $A[l_s, l_t] = \Delta I$.

5 Mechanistic Analysis for Knowledge–Circuit Misalignment

In this section, we propose the knowledge–circuit misalignment hypothesis to account for the observed Knowing–Using Gap through mechanistic interpretation. We provide interventional evidence consistent with the hypothesis that, after fine-tuning, answer-relevant representations can be recovered from hidden states but are not always routed through computation-effective layers. This misalignment of knowledge storage and computation results in the generalization failure, but can be largely recovered by manually relocating knowledge to the correct layers.

Knowledge–circuit misalignment hypothesis. A Knowing–Using Gap is driven by a spatial misalignment of knowledge storage and reasoning circuits: fine-tuning first encodes new facts in easy-to-fit storage states for memorization (often early or very late layers) that support direct recall, but are not reliably routed into the effective positions that are causally required for multi-step reasoning, often in mid-layer computation as shown in previous literature and our experiment in Figure 5.

This hypothesis predicts that (1) after memorization saturates, there should exist *off-path* representations that already contain injected information but do not yield correct reasoning end-to-end; and (2) explicitly relocating those representations into effective locations should immediately increase downstream use. We test these predictions with several causal intervention tools, mainly self-patching.

5.1 Self-Patching

Self-patching tests knowledge–circuit alignment through causal interventions: if a fact is already represented somewhere inside the model, can the target reasoning prompt use it when that represen-

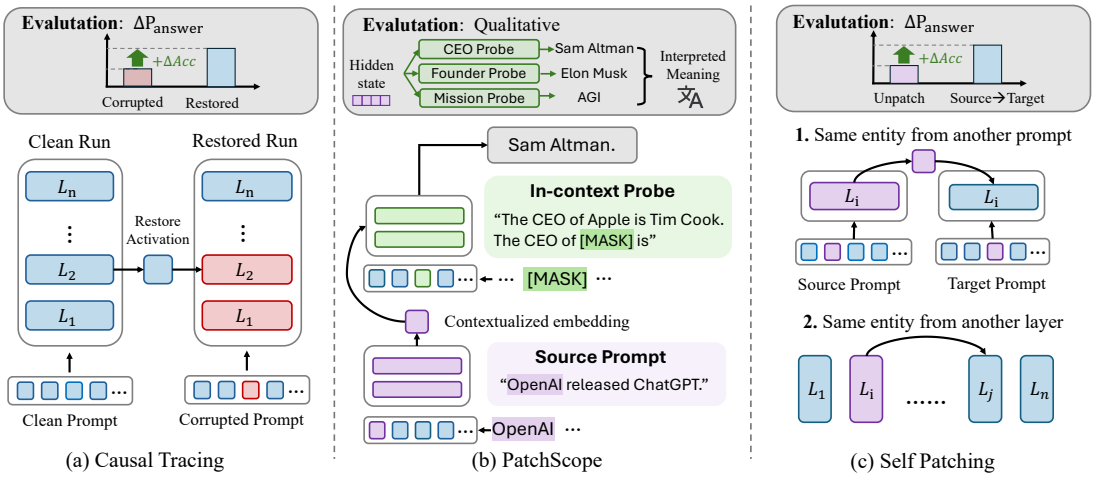


Figure 3: **Method Comparison.** *Causal tracing* corrupts a run to identify causally relevant locations. *Patchscope* interprets a hidden state by decoding it into natural language. *Self-patching* tests the effect position by swapping its internal representation across layers and contexts.

tation is routed through a candidate computation layer?

Let M be an L-layer transformer and let $h_t^l(P)$ denote the residual-stream state at layer l and token position t for prompt P. For an anchor substring E (e.g., the head entity), $T(P, E)$ denotes its token span. Given a source prompt P_s and a target prompt P_t, self-patching copies only the anchor representation from source layer l_s into target layer l_t, $\tilde{h}^{l_t}_{T(P_t,E)} \leftarrow h^{l_s}_{T(P_s,E)}$, then resumes the target forward pass. We measure the causal effect of this intervention as the change in an indicator function $I(\cdot, y^*)$, which evaluates the correctness of the generated output against y^* (e.g., using Exact Match or Mean Reciprocal Rank) with $\Delta I = I(\tilde{M}(P_t), y^*) - I(M(P_t), y^*)$.

We scan all layer pairs (l_s, l_t) after fine-tuning. A positive cell ($\Delta I > 0$) has a concrete interpretation: layer l_s contains a representation that becomes useful when placed at computation layer l_t. This produces the permeation maps in Figure 4 and the recoverable headroom in Table 4. To rule out prompt-specific shortcuts, we also patch across contexts from memorization prompts P_{mem} into generalization prompts P_{gen}; the transferred states preserve the same layer-pair structure (Figure 6).

Comparison with similar methods. We compare the most similar causal intervention based mechanistic interpretation method in Fig 3. Unlike causal tracing, self-patching does not require a "clean" correct trajectory and is therefore applicable to failed generalization cases. Unlike PatchScope, it focuses on the accuracy of target prompts instead of interpreting the source prompt with auxiliary decoding prompts; it evaluates whether the representation can unlock the downstream computation when routed into the model.

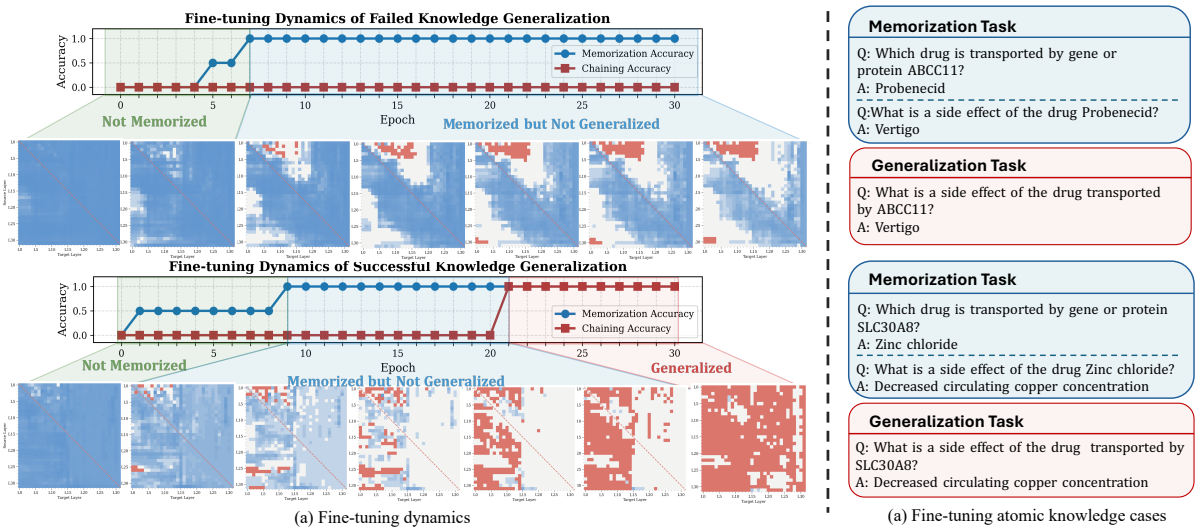
5.2 Knowledge Dynamics in Fine-Tuning

We first trace the knowledge permeation of two atomic knowledge during fine-tuning to understand *after a fact is memorized, when, where and how does it become accessible to the model's pre-existing reasoning logic.* We save checkpoints each epoch and perform layer-to-layer self-patching at the head-entity positions E_{head} on P_{gen}, producing a heatmap over $(l_{\text{src}}, l_{\text{tgt}})$ at each step.

Before memorization, patching rarely helps as indicated by the all-blue map: injected knowledge is not yet available internally at all, thus swapping can not help.

On the moment in training curve when two facts are memorized, we can observe clear *off-diagonal* red region appear in Figure 4 that says the needed information is already stored in certain layers and can take effect if routed into particular target layers. However, the model still fails end-to-end in the natural fine-tuning process: diagonal cells remains blue (fail with no intervention).

As fine-tuning continues, the red region expands wider towards the heatmap diagonal, indicating the gradual permeation of knowledge into more layers. In successful cases, the red region covers the diagonal line, which means the natural emergence of generalization from fine-tuning. In failed cases, while still expanding, the red region halts before reaching



(a) Fine-tuning dynamics

(a) Fine-tuning atomic knowledge cases

Figure 4: **Permeation dynamics of know–use transition, aligned with fine-tuning process.** Cell $(l_{\text{src}}, l_{\text{tgt}})$ measures whether patching the head-entity representation from l_{src} to l_{tgt} increases generalization accuracy, with red indicating full recovery, blue for no effect, and white for partial gain. **Top:** a natural failed instance. The red region indicates the position of knowledge storage, which permeates but halts before reaching the diagonal, indicating generalization failure. **Bottom:** a successful instance, patch-effective regions expand and cover diagonal before training stabilizes.

the diagonal. This is due to the fine-tuning nature that after memorization where $M(P)$ matches y^*, loss will diminish soon and become too small to drive further gradient updates for internal change. The model will stuck and fail to generalize unless manually relocating the representation, as shown in the next section. This knowledge permeation dynamics aligns well with the training phases in time, which validates the prediction from our hypothesis: knowledge is stored but not used until the representation becomes available in locations effective for the reasoning computation.

5.3 Manually Trigger Generalization

We then quantify the *oracle upper bound* at convergence to see how much downstream use can be activated by relocating representations. Since self-patch do not introduce new information but only relocates existing representations, the substantial improvement in downstream use after patching indicates that the generalization failure is not due to the absence of knowledge but rather a misalignment between storage and computation. We systematically fine-tune on different architectures, scales, and knowledge domains, and perform self-patching at convergence to measure the recoverable headroom. Experiments are computed with 1,000 samples to avoid randomness, detailing in Appendix B.

Table 4 shows that memorization saturates near-perfectly across models and domains, yet downstream use without intervention remains low, especially for chaining. Oracle self-patching yields large and consistent improvements along all architectures, model scales and knowledge domains. Chaining accuracy lifts by 1.5–6× in every cell, and the smaller know-use gap in intersection is nearly eliminated. The uniform improvement again points to a structural origin rather than an random artifact, showing causal evidence for the knowledge–circuit misalignment hypothesis.

Location of most effective patching. We further draw the spatial positions with most effective patching in Figure 5, revealing a clear two-cluster pattern: one cluster of source layers in early layers and another cluster in late layers, both patching into a middle cluster of target layers. While the later one is intuitive (Ghandeharioun et al., 2024; Biran et al., 2024) since moving latter representation backwards inherently brings in enriched information, the other cluster is surprising: moving from early layers into middle layers also triggers the generalization ability. This means that it is not that the need information emerges too late – the model can even succeed skipping several layers. Instead, the information is already stored in both early and late layers, but fail to align with the computation in middle layers, which is the bottleneck for use.

Model	STaRK-Prime					STaRK-MAG				
	Mem.	Chaining		Intersection		Mem.	Chaining		Intersection	
		w/o	pat.	w/o	pat.		w/o	pat.	w/o	pat.
Qwen-2.5-1.5B	0.998	0.078	0.440	0.793	0.987	0.991	0.046	0.286	0.928	0.994
Qwen-2.5-3B	0.997	0.114	0.542	0.798	0.986	0.986	0.052	0.288	0.958	0.994
Qwen-2.5-7B	0.996	0.124	0.504	0.774	0.956	0.982	0.068	0.310	0.976	0.994
LLaMA-3.2-1B	0.994	0.102	0.316	0.874	0.975	0.975	0.082	0.182	0.982	0.994
LLaMA-3.2-3B	0.993	0.126	0.404	0.815	0.969	0.974	0.064	0.240	0.982	1.000
LLaMA-3.1-8B	0.986	0.182	0.458	0.795	0.921	0.975	0.072	0.222	0.982	0.994

Table 4: **Oracle self-patching recovers downstream use across models, architectures, and knowledge domains.** "pat." denotes head-entity self-patching at the most effective layer, averaged over 1000 injected facts. Full 95% Wilson score confidence intervals are in Appendix 12.

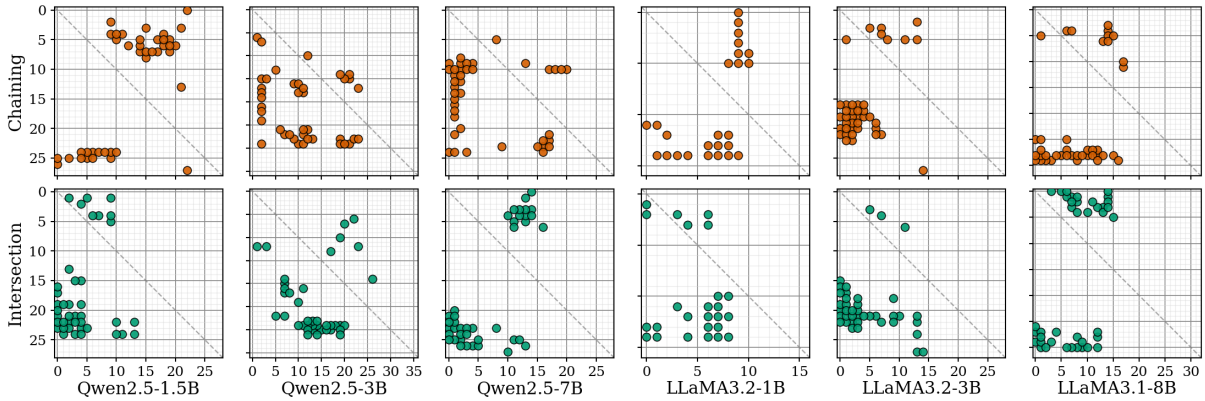


Figure 5: **Effective patch locations concentrate into two clusters.**

Table 5: **Token-position ablation.** McNemar's test on patching at different positions against random baseline.

Position	Mean	p-value	Significance
Random	0.1359	–	–
<BOS>	0.0485	1.0000	n.s.
<EOS>	0.4029	0.0000	***
First Relation	0.1990	0.0235	*
Second Relation	0.1990	0.0121	*
Entity	0.6408	0.0000	***

Moreover, relocating to late layers is useless as shown in the empty space at right, which can be illustrated by the three-step theory in previous work that the reasoning stream has moved to the last token in late layers (Geva et al., 2023).

5.4 Ablations and Sanity Checks

We perform several ablations to rule out other potential explanations such as position, prompting and perturbation artifacts.

Token-position Ablations. We experimentally prove that injected factual information is tightly tied to entity mentions but not position-agnostic. Table 5 shows that patching at the head-entity position yields the largest and most significant improvements, far exceeding random and <BOS>. The <EOS> position exhibits a secondary signal, consistent with late positions aggregating information relevant for generation.

Control experiment on prompting and perturbation.

Table 6 combines two complementary ablations on prompting and representation. CoT prompting improves chaining but remains far below self-patching, and sometimes even degrades intersection performance, and irrelevant patching that uses representation from unrelated fact still lags largely behind patching *correct* fact's representation.

Detect knowledge existence. To rule out the chance that gains are driven by prompt artifacts or noises, we further cross-context patch the E_{head} representation from P_{mem} which contains knowledge representation to P_{gen}. Figure 6 shows strong correlation in the effective layer-pair patterns from both contexts, supporting that self-patching transfers injected-knowledge representations similar to

Model	Chaining				Intersection			
	w/o pat.	CoT	Irrelevant pat.	Self pat.	w/o pat.	CoT	Irrelevant pat.	Self pat.
Qwen-2.5-1.5B	0.078	0.132	0.150	0.440	0.793	0.243	0.873	0.987
Qwen-2.5-3B	0.114	0.288	0.184	0.542	0.798	0.387	0.856	0.986
Qwen-2.5-7B	0.124	0.390	0.194	0.504	0.774	0.398	0.890	0.956

Table 6: **Prompting and perturbation ablations.** Direct/no patching and self-patching are shared anchors; CoT tests prompt effects, and irrelevant patching controls for generic activation perturbations.

Model	Mem.	Chaining			Intersection		
		No Pat.	Fixed	Oracle	No Pat.	Fixed	Oracle
Qwen-2.5-1.5B	0.998	0.078	0.349	0.440	0.793	0.942	0.987
Qwen-2.5-3B	0.997	0.114	0.435	0.542	0.798	0.943	0.986
Qwen-2.5-7B	0.996	0.124	0.409	0.504	0.774	0.914	0.956
LLaMA-3.2-1B	0.994	0.102	0.252	0.316	0.874	0.947	0.975
LLaMA-3.2-3B	0.993	0.126	0.321	0.404	0.815	0.926	0.969
LLaMA-3.1-8B	0.986	0.182	0.375	0.458	0.795	0.886	0.921
Average	—	0.121	0.357	0.444	0.808	0.926	0.966

Table 7: **Fixed heuristic vs. oracle self-patching.** "Fixed" uses two predetermined layer pairs (late→mid, early→mid) per model architecture, requiring no per-instance search.

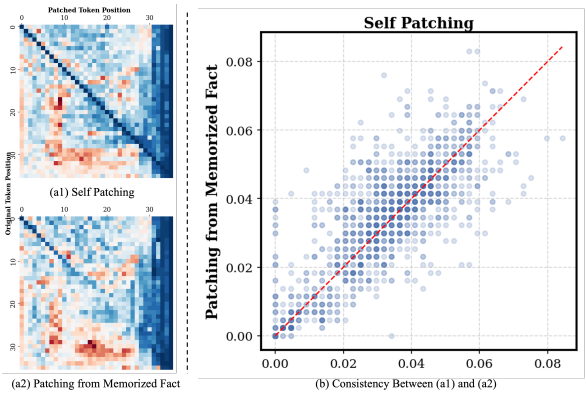


Figure 6: **Cross-context consistency.**

P_{mem} rather than some unknown noises.

5.5 From Diagnosis to Practical Heuristic

While oracle self-patching shows large headroom, it is a diagnostic tool rather than a practical method. A natural question is whether the observation in Figure 5 can be exploited for a practical strategy.

Fixed heuristic. Based on the concentration of effective patches into two clusters (late→mid and early→mid), we define a simple strategy using only two predetermined layer pairs per model architecture: (i) source at $\sim 0.8L$ targeting $\sim 0.5L$, and (ii) source at $\sim 0.1L$ targeting $\sim 0.5L$, where L is the total number of layers. These pairs require no per-instance search.

Results. Table 7 shows that the fixed heuristic recovers 58–75% of oracle headroom across all models and tasks, confirming that the knowing–using gap can be partially bridged with a practical strategy. The remaining gap between the fixed and oracle results reflects instance-specific variation in where knowledge is stored, motivating future work on adaptive alignment methods.

6 Conclusion

We identified the Knowing–Using Gap: fine-tuned LLMs memorize injected facts yet fail to use them in multi-hop reasoning. Using self-patching as a causal diagnostic, we trace this failure to *knowledge–circuit misalignment*—knowledge is encoded in storage-oriented early or late layers but does not permeate into the mid-layer circuits required for reasoning. The gap is thus a routing problem, not a capacity one—knowledge resides in the model, just not where reasoning happens. It is also partially reversible: a heuristic recovers 58–75% of oracle headroom, and the effect replicates across models, scales, and domains. Together, these results recast fine-tuning's generalization failure as a tractable alignment problem, and point to alignment-aware training as a principled path forward.

References

Alan Agresti and Brent A. Coull. 1998. Approximate is better than "exact" for interval estimation of binomial proportions. *The American Statistician*, 52(2):119–126.

Guillaume Alain and Yoshua Bengio. 2017. Understanding intermediate layers using linear classifier probes. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.3, knowledge capacity scaling laws. In *The Thirteenth International Conference on Learning Representations*.

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Comput. Linguistics*, 48(1):207–219.

Lukas Berglund, Meg Tong, Maximilian Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The reversal curse: Llms trained on "a is b" fail to learn "b is a". In *The Twelfth International Conference on Learning Representations*.

Eden Biran, Daniela Gottesman, Sohee Yang, Mor Geva, and Amir Globerson. 2024. Hopping too late: Exploring the limitations of large language models on multi-hop queries. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14113–14130.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2024. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*, 11:283–298.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 8493–8502. Association for Computational Linguistics.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. 2025. Scaling and evaluating sparse autoencoders. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Samuel J. Gershman, Ila Fiete, and Kazuki Irie. 2025. Key-value memory in the brain. *CoRR*, abs/2501.02950.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12216–12235.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. 2024. Patchscopes: A unifying framework for inspecting hidden representations of language models. In *International Conference on Machine Learning*, pages 15466–15490. PMLR.

Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. 2024. Model editing at scale leads to gradual and catastrophic forgetting. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 15202–15232.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. 2024. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Ziyue Li, Chenrui Fan, and Tianyi Zhou. 2025. Where to find grokking in LLM pretraining? monitor memorization-to-generalization without test. *CoRR*, abs/2506.21551.

Ziming Liu, Ouail Kitouni, Niklas Nolte, Eric J. Michaud, Max Tegmark, and Mike Williams. 2022. Towards understanding grokking: An effective theory of representation learning. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*.

John X Morris, Chawin Sitawarin, Chuan Guo, Narine Kokhlikyan, G Edward Suh, Alexander M Rush, Kamalika Chaudhuri, and Saeed Mahloujifar. 2025. How much do language models memorize? *arXiv preprint arXiv:2505.24832*.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. 2023. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

nostalgebraist. 2020. interpreting GPT: the logit lens. Accessed: 2025-12-29.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann,

Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, and 7 others. 2022. In-context learning and induction heads. *CoRR*, abs/2209.11895.

Oded Ovadia, Menachem Brief, Moshik Mishaeli, and Oren Elisha. 2024. Fine-tuning or retrieval? comparing knowledge injection in llms. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 237–250.

Vedant Palit, Rohan Pandey, Aryaman Arora, and Paul Pu Liang. 2023. Towards vision-language mechanistic interpretability: A causal tracing tool for BLIP. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023 - Workshops, Paris, France, October 2-6, 2023*, pages 2848–2853. IEEE.

Kiho Park, Yo Joong Choe, and Victor Veitch. 2024. The linear representation hypothesis and the geometry of large language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. 2022. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*.

Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. 2024. Fine-tuning enhances existing mechanisms: A case study on entity tracking. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Heydar Soudani, Evangelos Kanoulas, and Faegheh Hasibi. 2024. Fine tuning vs. retrieval augmented generation for less popular knowledge. In *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pages 12–22.

Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. 2024. Grokking of implicit reasoning in transformers: A mechanistic journey to the edge of generalization. *Advances in Neural Information Processing Systems*, 37:95238–95265.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2023. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Chris Wendler, Veniamin Veselovsky, Giovanni Monea, and Robert West. 2024. Do llamas work in english? on the latent language of multilingual transformers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 15366–15394. Association for Computational Linguistics.

Shirley Wu, Shiyu Zhao, Michihiro Yasunaga, Kexin Huang, Kaidi Cao, Qian Huang, Vassilis N. Ioannidis, Karthik Subbian, James Y. Zou, and Jure Leskovec. 2024. Stark: Benchmarking LLM retrieval on textual and relational knowledge bases. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Yunzhi Yao, Jizhan Fang, Jia-Chen Gu, Ningyu Zhang, Shumin Deng, Huajun Chen, and Nanyun Peng. 2025. Cake: Circuit-aware editing enables generalizable knowledge learners. *arXiv preprint arXiv:2503.16356*.

Yunzhi Yao, Ningyu Zhang, Zekun Xi, Mengru Wang, Ziwen Xu, Shumin Deng, and Huajun Chen. 2024. Knowledge circuits in pretrained transformers. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *The Twelfth International Conference on Learning Representations*.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. 2023. Mquake: Assessing knowledge editing in language models via multi-hop questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15686–15702.

A Dataset details

A.1 Construction pipeline

We present an automated pipeline designed to sample facts from the knowledge graph and generate QA pairs for the dataset. Two distinct methods are designed for the generation process: an LLM-based approach to ensure linguistic diversity, and a template-based approach to maintain structural consistency. The procedure comprises the following steps:

1. **Sample Valid Paths** For each type of generalization task, we manually design a list of valid meta paths to guide the sampling process and generate meaningful and reasonable questions. A meta path defines a sequence of entity types and relation types that outline the structure of the knowledge to be used for the task. For instance, a meta-path for chaining task may be denoted as (anatomy $\rightarrow$ expression Present $\rightarrow$ gene/protein $\rightarrow$ target $\rightarrow$ drug). Concrete paths (i.e., comprising specific entity

sequences) matching the meta paths are sampled from the knowledge graph accordingly, e.g., (female reproductive system $\rightarrow$ expression present $\rightarrow$ SLC12A6 $\rightarrow$ target $\rightarrow$ potassium chloride).

2. **Generate Memorization Tasks** We randomly sample a path matching a meta path from the first step, and decompose it into a list of fact triplets, the minimal units of knowledge in our setting, as well as the material for finetuning LLMs. For each fact triplet, we prompt an AI assistant or use a predefined template to generate the memorization task. Specifically, for intersection tasks, some extra noise facts related to both head entities are added to the list for task complexity.

3. **Generate Generalization Tasks** For the same sampled path, we use similar method to generate the corresponding generalization task question with AI assistant or templates.

This pipeline is automated, efficient, and scalable, offering two alternative methods for QA pair generation. The LLM-based method yields diverse and more natural QA pairs, while the template-based method ensures consistency and controllability for downstream interpretability studies.

A.2 Requirements validation

We validate that our dataset meets all 4 requirements mentioned in 3.1. **Scalability** is guaranteed by the 8 million distinct fact triplets in STaRK-Prime KG, from which we directly generate dataset samples from. For **task diversity**, we manually design 3 generalization tasks organizing support facts in different ways, and generate them from 53 meta paths covering 10 entity types and 18 relation types. And **real-world knowledge** in our dataset is grounded by the biomedical KG. For cross-domain validation, we additionally construct an analogous dataset from STaRK-MAG (academic knowledge graph) following the same pipeline with author–paper–field-of-study relations, generating 12,000 chaining and 4,390 intersection items.

To validate that our dataset contains **minimal prior knowledge** to the LLMs, we conducted an experiment to evaluate the novelty of the facts. Specifically, we randomly sampled 1,000 facts and their corresponding memorization tasks from the dataset and assessed the evaluate the zero-shot accuracy of pre-trained LLMs on these tasks without any fine-tuning. As results shown in Table 2, on both STARK-PRIME and STARK-MAG, all models score around or below 0.06, indicating that the models have little prior knowledge of the dataset samples.

B Experiment Setup Details

This section consolidates the experimental setup used by the phenomenon and mechanistic analyses in the main paper. Unless otherwise stated, experiments are run on a fixed 1,000-sample split of injected facts to reduce sampling noise; smaller counts only arise after task-specific filtering or when a diagnostic scan is intentionally run on a smaller subset for computational cost.

Knowledge-injection tasks. The default domain is STARK-PRIME, with STARK-MAG used for cross-domain replication. For the chaining experiments, each instance contains two support facts $E_1 \rightarrow r_1 \rightarrow E_2$ and $E_2 \rightarrow r_2 \rightarrow E_3$. The model is fine-tuned on the corresponding memorization queries and evaluated on the held-out compositional query $E_1 \rightarrow r_1 \rightarrow r_2 \rightarrow ?$, which requires recovering the bridge entity before predicting the final answer. Intersection uses the same memorization-then-use separation, but evaluates whether the model can identify the shared entity from multiple support facts and noise facts. Before patching, we filter out instances already answered correctly by the base model on either the memorization or generalization prompt, so the measured gains cannot be attributed to pre-existing knowledge leakage.

Training setup for the Knowing–Using Gap. For the temporal-lag and accuracy-gap measurements in Section 4.2, we fine-tune open-weight LLaMA and Qwen models on randomly sampled knowledge-injection tasks and track memorization and downstream-use accuracy at each checkpoint. The LoRA runs use AdamW with weight decay 0.01; the practical injection setting in the original setup uses batch size 10, learning rate 10^{-4}, and 50 epochs, long enough for direct memorization to saturate. The full fine-tuning comparison uses the same task split and evaluation protocol as the LoRA comparison. For model-scale and data-scale experiments, we keep the task construction fixed while varying either the base model size or the number of injected facts, and report both the final accuracy gap ΔA and the temporal lag ΔT.

LoRA hyperparameters for the main multi-fact

Table 8: Target prompt used in patchscope

meta path (key)	in-context prompt (value)
('disease', 'associated with', 'gene/protein')	cystic fibrosis->CFTR; sickle cell disease->HBB; Duchenne muscular dystrophy->DMD; disease name->
('effect/phenotype', 'associated with', 'gene/protein')	PTC bitter-tasting ability->TAS2R38; red-green color vision defect->OPN1LW; lactase persistence (adult lactose digestion)->LCT; phenotype name->
('disease', 'contraindication', 'drug')	G6PD deficiency->primaquine; asthma->propranolol; Parkinson's disease->metoclopramide; disease name->

runs. The consolidated LoRA configuration uses rank $r = 16$, $\alpha = 32$, dropout 0.05, and target modules $\{q, k, v, o, \text{gate}, \text{up}, \text{down}\}_{\text{proj}}$ across all transformer blocks. We use AdamW with default $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, weight decay 0.01, no warmup, and greedy evaluation decoding. The reproduction scripts use learning rate 2×10^{-4}, per-device batch size 1, gradient accumulation 8 (effective batch size 8), fp32 gradients, and random seed 42. Main multi-fact runs use the fixed 1,000-fact split unless a table explicitly reports a filtered evaluation count.

Permeation dynamics. For the training-dynamics heatmaps in Section 5.2, we save checkpoints at every epoch and run layer-to-layer self-patching at the head-entity token positions E_{head}. For each checkpoint, we scan source and target layer pairs $(l_{\text{src}}, l_{\text{tgt}})$ and record whether replacing the target residual-stream representation with the source representation improves the downstream answer. This diagnostic scan is run on 100 randomly sampled chaining tasks over 30 epochs with parameter-efficient tuning, which is sufficient to visualize when memorized representations become causally usable.

Oracle and fixed self-patching. For the converged recoverable-headroom results in Section 5.3, each instance is evaluated by scanning all layer pairs on the head-entity position in the generalization prompt and reporting the best-performing pair as the oracle diagnostic upper bound. The fixed heuristic in Section 5.5 uses only two predetermined pairs per model, $(\lfloor 0.82L \rceil, \lfloor 0.45L \rceil)$ and $(\lfloor 0.10L \rceil, \lfloor 0.45L \rceil)$, where L is the number of transformer layers. No per-instance search is used by the fixed heuristic. For STARK-MAG, we follow the same memorization, chaining, intersection, filtering, fine-tuning, and self-patching protocol as STARK-PRIME; the patching evaluation uses 500 MAG chaining instances and 166 MAG intersection instances after filtering, as detailed in Appendix C.

Controls and statistical reporting. The token-position ablation uses the same converged checkpoints but changes the patched position among random, <BOS>, <EOS>, relation tokens, and the entity token. Prompting and perturbation controls compare direct generation, CoT prompting, irrelevant-fact patching, and self-patching with the correct fact representation under the same exact-match scoring rule. For all aggregate proportions, Appendix F reports 95% Wilson confidence intervals; paired memorization–use gaps are tested with McNemar's test and an exact binomial test on discordant pairs, and the temporal-lag definition is checked under a τ, w sensitivity grid.

C Cross-Domain Replication Details (STaRK-MAG)

C.1 Domain comparison

Table 9 compares the two knowledge domains used in this work. STaRK-Prime covers biomedical entities (diseases, genes, drugs) with rich relational structure, while STaRK-MAG covers academic publications (authors, papers, fields of study). The domains differ in entity naming conventions (concise biomedical terms vs. verbose paper titles), relation types, and graph density, providing a meaningful test of cross-domain generality.

Table 9: Comparison of the two knowledge domains.

	STaRK-Prime	STaRK-MAG
Source	PrimeKG	Microsoft Academic
Domain	Biomedical	Academic
Entity types	Disease, Gene, Drug, Anatomy, Phenotype	Author, Paper, Field of Study
Relation types	18	5
Entity naming	Concise terms	Verbose titles
Chaining items	1,000	12,000
Intersection items	996	4,390

Table 10: Temporal Lag (ΔT) across different tasks on Llama-3.2-8B. Values denote Epochs (Mean $\pm$ Std).

Task	T_{mem}	T_{gen}	ΔT (Lag)
Chaining	7.23 ± 1.89	11.83 ± 5.41	$+4.60$
Intersection	6.04 ± 1.32	7.16 ± 6.96	$+1.12$

C.2 Training dynamics

Figure 7 shows epoch-by-epoch memorization and generation accuracy for the Qwen MAG training runs. **Chaining** exhibits the starkest decoupling: memorization reaches >0.95 by epoch 15 while generation remains near zero throughout—a pattern that holds across Qwen model scales and mirrors the biomedical domain. **Intersection** shows memorization and generation rising in tandem, reaching similarly high final values—consistent with the small gap observed in both domains.

D Scaling Results on Knowing–Using Gap

We test the knowing-using gap across different model scales and data scales, with results in Appendix.

Figure 2 shows that increasing model size does not eliminate the temporal lag ΔT. Moreover, increasing the number of injected facts tends to widen the final accuracy gap $\Delta A(\mathcal{T})$, even when direct recall remains strong, indicating that scaling storage does not directly translate into proportional gains in usable reasoning. Table 10 gives the corresponding LLaMA-3.2-8B lag estimates for chaining and intersection: chaining shows a larger delay, while intersection remains much closer to memorization time.

E More results on knowledge-circuit misalignment.

We use PatchScope to interpret the hidden state in the memorization prompt. We use the in-context prompt shown in Table 8, and test the head-entity position. Results in Figure 8 show that for early-layer knowledge storage, the detectable place in LLaMA is lower than that in Qwen across model scales, suggesting that LLaMA tends to store knowledge in early to middle layers.

Table 11: **Control patching experiments.** for LLaMA-3.2-3b

	Chain.	Intersec.
No-Patching	0.126	0.815
Irrelevant-Patching	0.207	0.915
Self-Patching	0.404	0.969

We also report the controlled experiment for self-patching for the LLaMA architecture same size as used in main text, as shown in Table 11. The control confirms that self-patching yields the strongest recovery on both chaining and intersection, while irrelevant patching remains below the knowledge-specific intervention.

F Statistical Significance and Confidence Intervals

We report 95% Wilson score confidence intervals for all main results. Wilson intervals are preferred over normal approximation for proportions near 0 or 1 (Agresti and Coull, 1998). Memorization is evaluated on $n = 1000$ facts; chaining and intersection on $n = 500$ and $n = 1000$ respectively. The per-instance Knowing–Using Gap (Mem.−Chain.) is significant for all models under McNemar's test on paired 0/1 outcomes ($p < 10^{-50}$ for every model). The gap magnitude is fully described by the difference of proportions Mem.−Chain., which is itself the McNemar effect-size statistic and is shown in the right column. An exact binomial test on the discordant pairs gives the same conclusion as McNemar. Spearman correlation between model size and gap magnitude is $r = -0.90$, $p = 0.015$. For temporal-lag sensitivity, recomputing ΔT on Qwen-2.5-3B chaining over $\tau \in \{0.9, 0.95, 1.0\}$ and $w \in \{1, 2, 3\}$ yields lag values within ± 1.0 epoch of the default ($\tau = 1.0$, $w = 2$) report; the qualitative pattern $T_{\text{use}} > T_{\text{mem}}$ holds in all 9 grid cells.

G More cases on permeation dynamics

Fig 9 and fig. 10 demonstrate more cases of the permeation dynamics of transitions from memorization to generalization on LLaMA-3.1-8B and

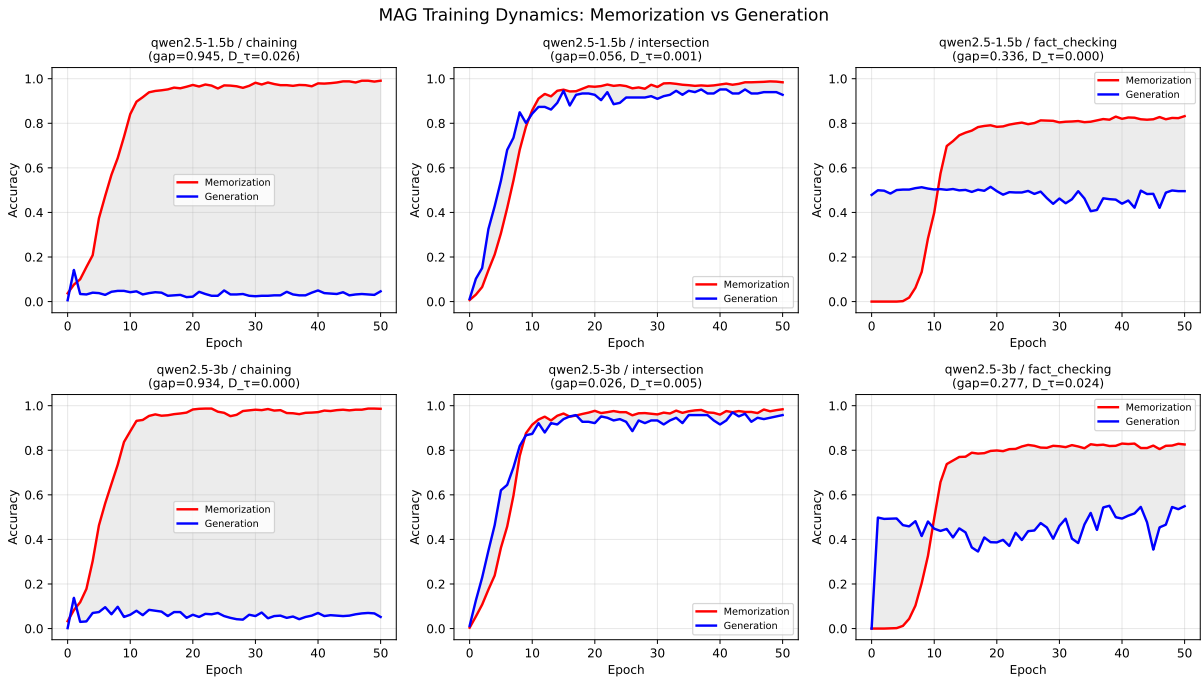


Figure 7: **MAG training dynamics across tasks and model scales.** Each panel shows memorization (red) and generation (blue) accuracy over 50 training epochs; the shaded region visualizes the instantaneous KU gap. Chaining exhibits a large, persistent gap; intersection shows near-parallel learning curves.

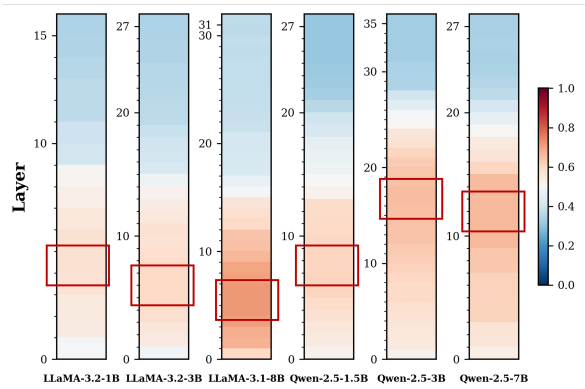


Figure 8: Patchscope views of memorization-related layers at the head-entity position.

Model	Mem.	Chain.	Intersec.	Gap
Qwen-2.5-1.5B	$0.998^{+0.002}_{-0.004}$	$0.078^{+0.025}_{-0.020}$	$0.793^{+0.032}_{-0.038}$	0.920
Qwen-2.5-3B	$0.997^{+0.002}_{-0.004}$	$0.114^{+0.030}_{-0.024}$	$0.798^{+0.032}_{-0.037}$	0.883
Qwen-2.5-7B	$0.996^{+0.003}_{-0.005}$	$0.124^{+0.031}_{-0.026}$	$0.774^{+0.034}_{-0.039}$	0.872
LLaMA-3.2-1B	$0.994^{+0.004}_{-0.006}$	$0.102^{+0.028}_{-0.023}$	$0.874^{+0.026}_{-0.032}$	0.892
LLaMA-3.2-3B	$0.993^{+0.004}_{-0.006}$	$0.126^{+0.031}_{-0.026}$	$0.815^{+0.031}_{-0.036}$	0.867
LLaMA-3.1-8B	$0.986^{+0.006}_{-0.009}$	$0.182^{+0.035}_{-0.031}$	$0.795^{+0.032}_{-0.038}$	0.804

Table 12: 95% Wilson score confidence intervals for Table 4.

Qwen-2.5-7B respectively. The patterns are all coherence with results in 5.2 where patch-effective regions covering diagonal indicates the emergence of direct chaining ability.

H Limitations and Responsible Research Details

H.1 Limitations

Self-patching is deliberately constrained: we intervene at a fixed anchor position and move representations across layers. Injected knowledge may also distribute across multiple positions or be redundantly encoded, so our estimates may understate the full recoverable headroom.

Our oracle best-pair results should be interpreted as a diagnostic upper bound. While the fixed heuristic in Section 5.5 demonstrates practical feasibility, developing fully adaptive non-oracle alignment strategies remains an important direction.

Our mechanism is mainly diagnostic: we can identify and partially repair the misalignment post hoc, but we do not yet offer an early-training signal that forecasts which facts will fail to generalize. Developing such a predictive metric would enable proactive interventions during fine-tuning.

Finally, our mechanistic analysis operates at the token-layer level. Finer-grained localization to specific attention heads or MLP sublayers could further refine the knowledge–circuit misalignment hy-

pothesis and inform more targeted interventions.

H.2 Reproducibility and Compute

Code, data, and reproduction instructions are included in the anonymous release at https://anonymous.4open.science/r/Mem2Gen-71FF. Appendix B reports the main data splits, filtering protocol, hyperparameters, evaluation rules, and statistical tests used to reproduce the experimental results.

Experiments were run on a server with NVIDIA A800 GPUs, each with 81,920 MiB of memory, an Intel 128-core CPU, and 512 GB system memory. A typical primary experiment used 8 GPUs. Depending on model scale and experiment type, the wall-clock time for a single run ranged from about 5 hours to 1 day.

H.3 Assets, Licenses, and Released Artifacts

The study adapts the STARK benchmark (Wu et al., 2024), specifically STARK-PRIME and STARK-MAG; the public STaRK repository is released under the MIT License.[1] We use open-weight Qwen2.5 models under their public licenses: Qwen2.5-1.5B and Qwen2.5-7B under Apache 2.0, and Qwen2.5-3B under the Qwen Research/Tongyi Qianwen Research terms.[2] We use LLaMA-3.1 and LLaMA-3.2 models under the Meta Llama Community License terms.[3]

We release the derived memorization-to-generalization QA data and accompanying code as new artifacts under the MIT License in the anonymous repository. The released data are generated from knowledge-graph facts and templates or AI-assisted wording; no consent-bearing human-subject data are collected for this work.

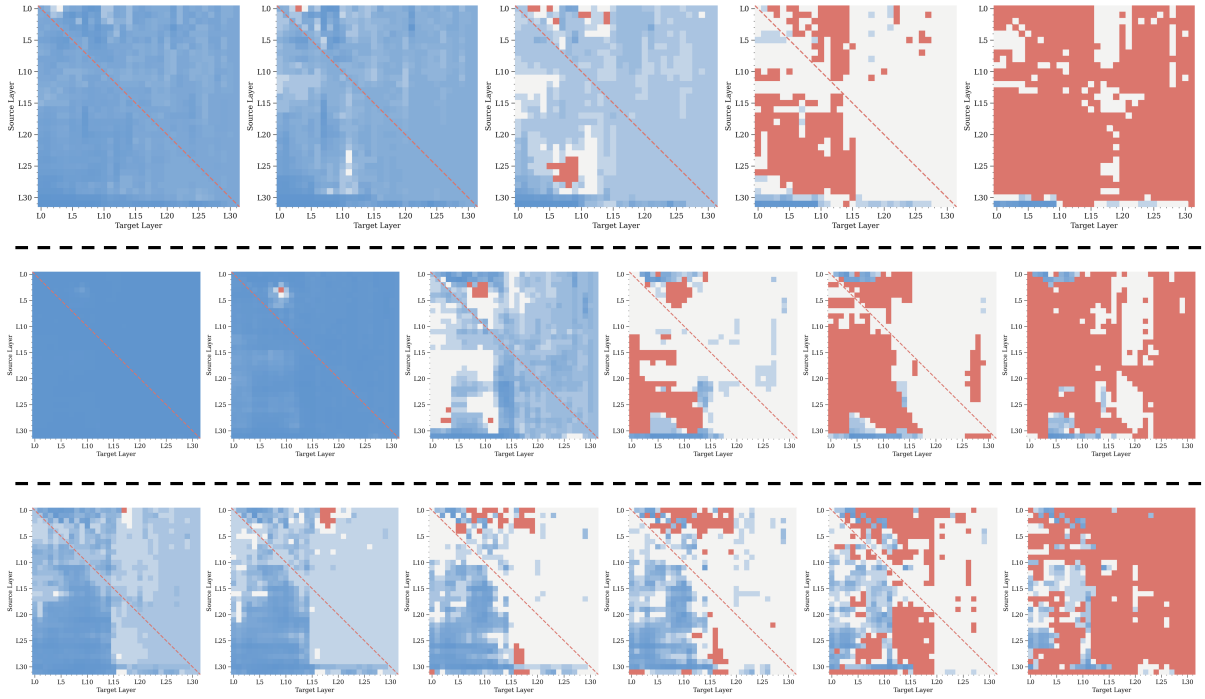
H.4 Ethics, Broader Impacts, and Safeguards

The authors reviewed the NeurIPS Code of Ethics and believe the work conforms to it. The work does not involve crowdsourcing, human-subject experiments, participant compensation, or IRB-style review requirements.

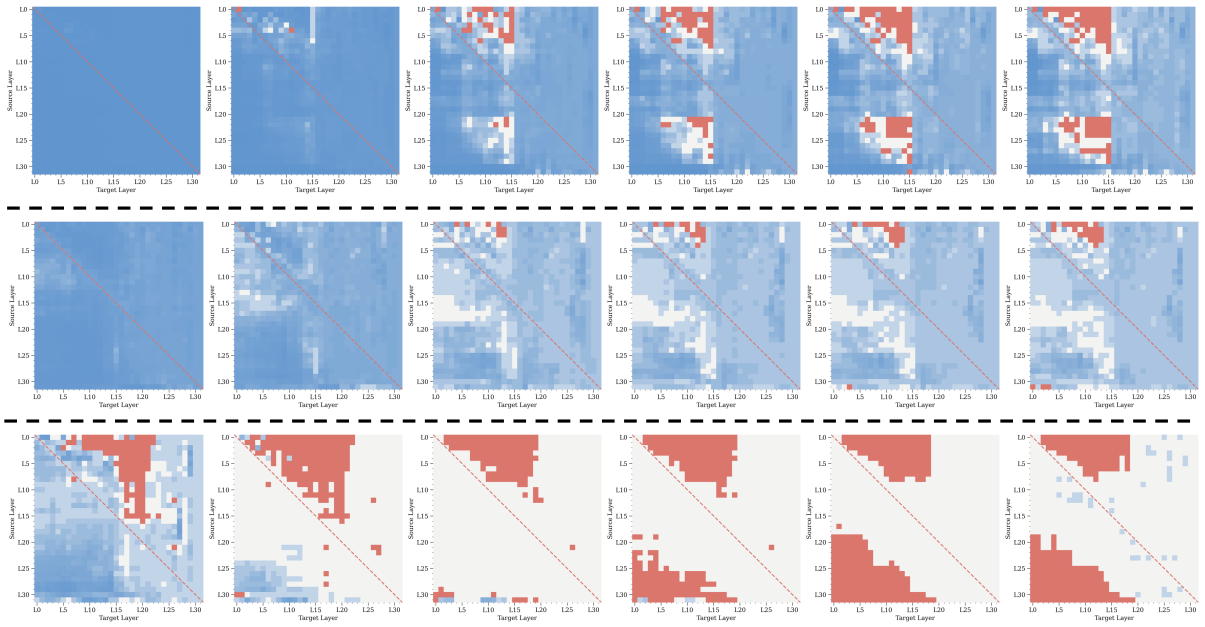
[1] https://github.com/snap-stanford/stark

[2] https://qwen2.org/qwen2-5/

[3] https://huggingface.co/meta-llama/Llama-3.1-405B/blob/main/LICENSE and https://huggingface.co/meta-llama/Llama-3.2-3B/blob/main/LICENSE.txt

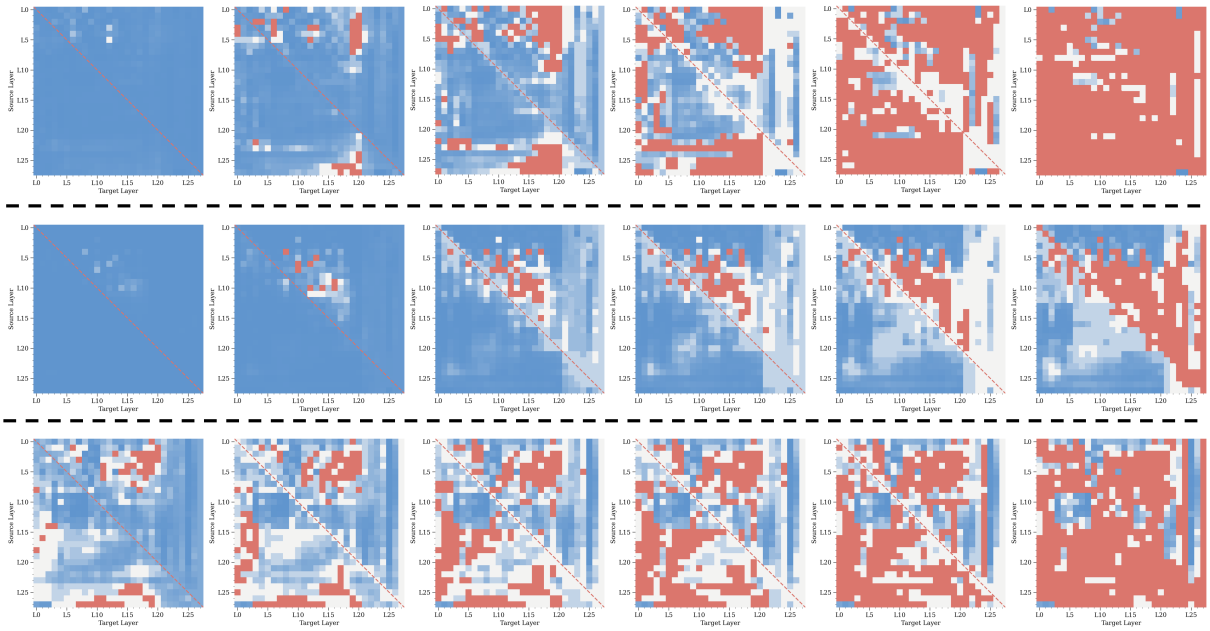


(a) success cases

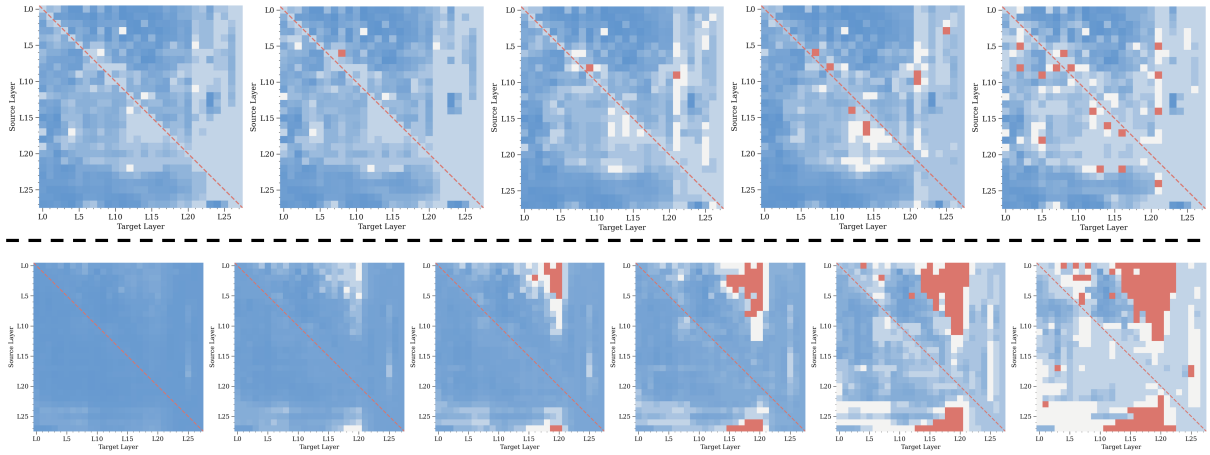


(b) failure cases

Figure 9: **More permeation dynamics cases on LLaMA-3.1-8B.**



(a) success cases



(b) failure cases

Figure 10: **More permeation dynamics cases on Qwen-2.5-7B.**