TOWARDS MECHANISTICALLY UNDERSTANDING WHY MEMORIZED KNOWLEDGE FAILS TO GENERALIZE IN LARGE LANGUAGE MODEL FINETUNING

Anonymous authors
Paper under double-blind review

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 with self-patching, an adaptation of activation patching that scans all layer pairs at every fine-tuning checkpoint. 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. Experiments are done cross-domain for the robustness of this finding. Building on this diagnosis, we propose layer-wise representation self-distillation (LRSD) that aligns knowledge representation from late storage layer to middle layer. LRSD keeps improving generalization after fine-tuning saturates and nearly doubles multi-hop chaining accuracy on Qwen, while leaving memorization intact. 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 in principle allows reuse by the model's existing reasoning capabilities.

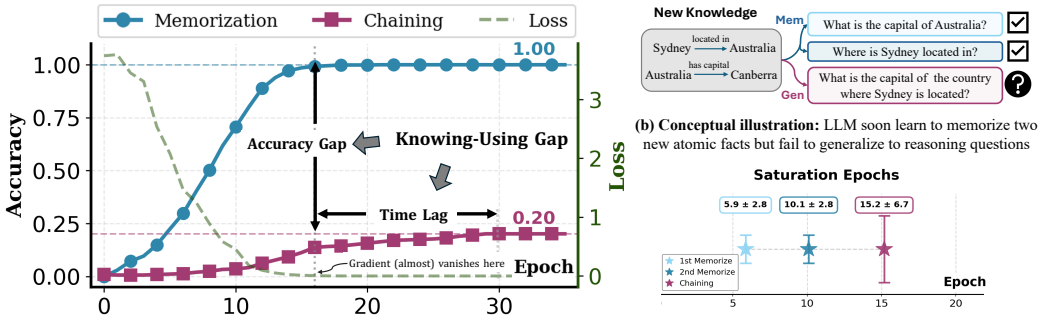
Despite sufficient capacity to fit new data Morris et al. (2025); Allen-Zhu & Li, LLMs exhibit a "remembering but not using" failure Ovadia et al. (2024); Soudani et al. (2024); Zhong et al. (2023); Cohen et al. (2024); Berglund et al. as shown in Figure 1: models can memorize new facts (e.g., *"Sydney 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

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(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.**

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 adapt activation patching (Meng et al., 2022; Ghandeharioun et al., 2024; Zhang & Nanda, 2024) into *self-patching*, which copies the hidden representation of an entity from one layer into another layer of the same fine-tuned model and measures whether the correct answer is recovered. By scanning all layer pairs at every fine-tuning checkpoint, 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. Finally, we turn this diagnosis into a training objective. Layer-wise representation self-distillation (LRSD) aligns, during fine-tuning, the head-entity representation in a middle computation layer with that in a deeper storage layer. With one layer pair fixed per model and no test-time intervention, LRSD improves chaining accuracy in all four models tested and nearly doubles it on Qwen2.5 (§3.5), 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 design a **self-patching** scan that traces **knowledge permeation** during fine-tuning, locating where the knowledge becomes *causally usable* throughout fine-tuning steps.
- We propose the **knowledge–circuit misalignment hypothesis** and provide mechanistic evidence that manually relocating memorized representations recovers generalization across domains and architectures.
- We turn the diagnosis into **layer-wise representation self-distillation (LRSD)**, a fine-tuning method that aligns storage and computation layer representation to improves generalization, proving its potential to guide the design of generalizable knowledge injection methods.
- We release a specialized **Memorization-to-Generalization dataset** for evaluating multi-hop reasoning on injected knowledge.

Task	Split	Question	Answer
Chaining	Train	Which gene or protein is expressed in embryo?	**IGFBP3**
		Which drug targets the gene or protein IGFBP3?	**Mecasermin**
	Test	Which drug targets the genes or proteins that are expressed in embryo?	**Mecasermin**
Intersection	Train	Which gene or protein acts as a carrier for the drug Acitretin?	**ALB**
		Which gene or protein acts as a carrier for the drug Propiolactone?	**ALB**
		What is a side effect of the drug Acitretin? *(distractor)*	**Tinnitus**
	Test	What gene or protein is a carrier for both Acitretin and Propiolactone?	**ALB**

		Knowledge graph		# Tasks		Prior recall
Source	**Domain**	Entities	Relations	Chaining	Intersection	
STARK-PRIME	Biomedical	22,193	50,653	11,997	10,000	3.8–4.9%
STARK-MAG	Academic	55,149	50,092	12,000	4,390	5.0–6.0%

Table 1: **Tasks and knowledge sources.** *Top:* verbatim STARK-PRIME examples. Supporting facts are injected as memorization QA pairs (Train); the generalization question (Test) is held out. Chaining composes two facts via a bridge entity; intersection finds the entity shared by two facts, ignoring distractors. Colors mark entity roles: query, bridge, answer, distractor. *Bottom:* task and fact statistics; prior recall (memorization accuracy, range over six base models; per model in Table 5) is low, confirming the facts are novel and ruling out knowledge leakage from pre-training.

2 THE KNOWING–USING GAP

2.1 DATA CONSTRUCTION AND EXPERIMENTAL SETTING

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) set to ensure the robustness of our findings across domains. The generation pipeline is detailed in Appendix A.

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".

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.

Knowledge Novelty Validation. To ensure the injected knowledge is genuinely novel, we first verified that pre-trained models score around or below 6% accuracy on 1,000 randomly sampled memorization tasks before any fine-tuning, on both STARK-PRIME and STARK-MAG (Table 5). Furthermore, during our patching experiments (§3), we explicitly filter out instances where the base

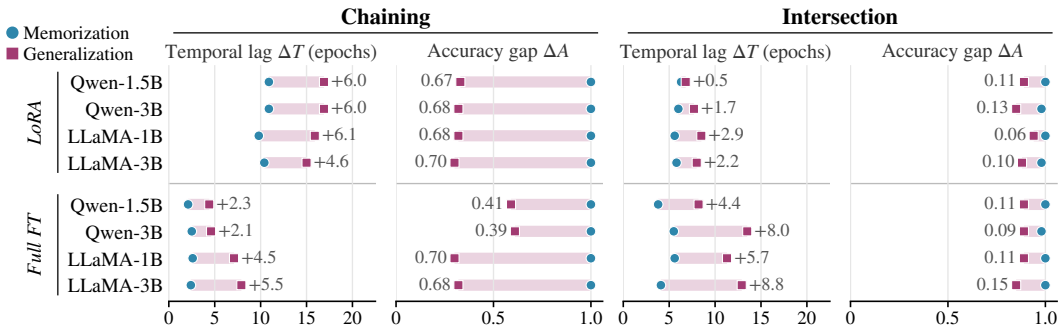


Figure 2: **The Knowing–Using Gap across tasks, fine-tuning schemes, and models**. Circles mark memorization, squares generalization; each bar and its value give the gap. $\Delta T = T_{\text{gen}} - T_{\text{mem}}$: epochs until all injected facts are memorized vs. generalized. $\Delta A = A_{\text{mem}} - A_{\text{gen}}$ at the final epoch. Each fact is injected in its own 30-epoch run; values average over runs (exact values in Table 7).

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 → 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.

2.2 Formalizing and Characterizing the 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

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

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:

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

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.

Figure 1 illustrates the Knowing–Using Gap under chaining: memorization reaches near-ceiling accuracy within a few epochs, while downstream use stays low for an extended period.

Figure 2 shows the ubiquity of this dissociation: memorization saturates early, while reliable downstream use is delayed and can remain substantially lower at convergence. *Across tasks*, chaining shows a two-dimensional gap: it requires substantially more training to become usable and still converges to a low use accuracy. Intersection is close to an "ideal" regime, with high use accuracy and, under LoRA, minimal lag, since its answer often aligns with the memorization answer. *Across fine-tuning schemes*, full fine-tuning memorizes substantially faster than LoRA, but this advantage does not consistently translate into faster or better downstream use: chaining still lags and stays at or below 0.61 accuracy, and intersection shows a markedly larger lag. *Across scales*, increasing model size does not eliminate the gap, and more injected facts tend to widen ΔA (Appendix C).

3 MECHANISTIC ANALYSIS FOR KNOWLEDGE–CIRCUIT MISALIGNMENT

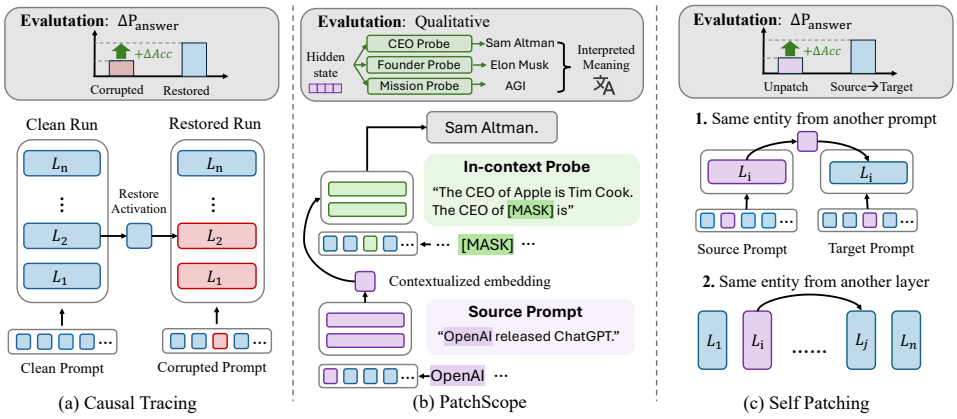


Figure 3: **Method Comparison.** *Causal tracing* corrupts a run to identify causally relevant locations. *Patchscope* interprets a hidden state by decoding it into natural language. *Self-patching* tests the effect position by swapping its internal representation across layers and contexts.

This section investigates the mechanism underlying the Knowing–Using Gap. We hypothesize that injected facts reside in hidden states but are not routed through the layers performing multi-hop reasoning, and show that relocating them substantially restores generalization. §3.5 turns this diagnosis into a fine-tuning objective.

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.

3.1 TRACING KNOWLEDGE PERMEATION WITH SELF-PATCHING

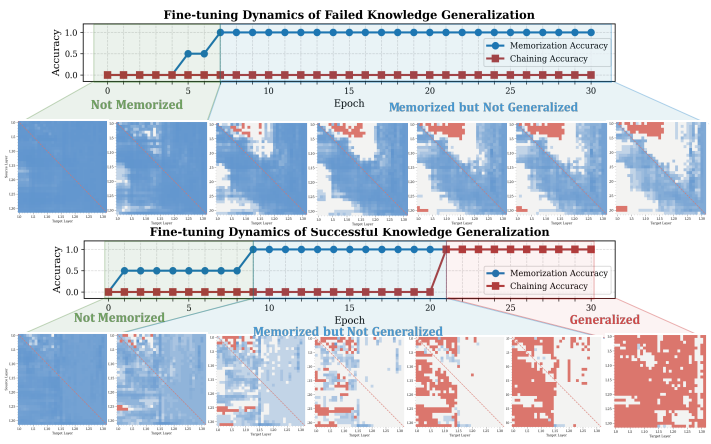
Self-patching is a form of activation patching (Meng et al., 2022; Zhang & Nanda, 2024; Ghandeharioun et al., 2024), adapted here to trace knowledge permeation (Algorithm 1). It copies the residual-stream states $h^{l_{\text{src}}}_{T_s}(P_s)$ of an L-layer model M at layer l_{src} over the tokens $T_s = T(P_s, E)$ of the head entity E in a source prompt P_s into layer l_{tgt} of a target prompt P_t; I scores the resumed run $\tilde{M}(P_t)$ against the answer y^* by exact match or mean reciprocal rank.

Algorithm 1: Self-patching scan

Input: model M, prompts P_s, P_t, anchor E, answer y^*, score I.
For each layer pair $(l_{\text{src}}, l_{\text{tgt}})$:
- ▷ Locate anchor tokens $T_s = T(P_s, E)$ and $T_t = T(P_t, E)$.
- ▷ Cache source state $z = h^{l_{\text{src}}}_{T_s}(P_s)$.
- ▷ Run P_t and replace $\tilde{h}^{l_{\text{tgt}}}_{T_t} \leftarrow z$.
- ▷ 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 $A[l_{\text{src}}, l_{\text{tgt}}] = \Delta I$.

Scanning layer pairs across training. Patching a multi-hop query onto itself ($P_s = P_t = P_{\text{gen}}$) only relocates the model's own representation and, unlike causal tracing (Figure 3), needs no correct run, so it applies to failed cases. In the $L \times L$ map of patched scores $I(\tilde{M}(P_t), y^*)$, the diagonal ($l_{\text{src}} = l_{\text{tgt}}$) is the unpatched model, and a high off-diagonal cell marks a representation at l_{src} usable once placed at l_{tgt}. Repeated each epoch, the maps trace permeation as high-score cells spreading toward the diagonal, and generalization as their arrival (Figure 4).



(a) Fine-tuning dynamics

Memorization Task

Q: Which drug is transported by gene or protein ABCC11?
A: Probenecid

Q:What is a side effect of the drug Probenecid?
A: Vertigo

Generalization Task

Q: What is a side effect of the drug transported by ABCC11?
A: Vertigo

Memorization Task

Q: Which drug is transported by gene or protein SLC30A8?
A: Zinc chloride

Q: What is a side effect of the drug Zinc chloride?
A: Decreased circulating copper concentration

Generalization Task

Q: What is a side effect of the drug transported by SLC30A8?
A: Decreased circulating copper concentration

(a) Fine-tuning atomic knowledge cases

Figure 4: **Permeation dynamics of knowledge 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.

3.2 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 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.

3.3 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 2 shows that memorization saturates near-perfectly across models and domains, yet downstream 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 2: **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.

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 2.2–6.2× 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.

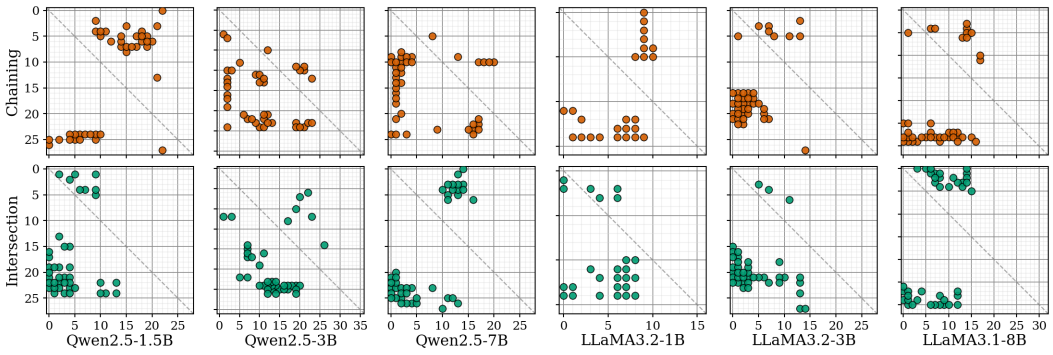


Figure 5: **Effective patch locations concentrate into two clusters.** Dots means the top-40 layer pairs that recover most generalizaiton failure on the whole dataset; y-axis is l_{src}, x-axis is l_{tgt}.

Location of most effective patching. In Figure 5, the most effective source layers form an early and a late cluster, both patching into middle target layers. The late cluster is intuitive (Ghandeharioun et al., 2024; Biran et al., 2024), as later representations carry enriched information; the early one is surprising: moved into middle layers, early representations also trigger generalization, even skipping several layers. Hence the needed information does not emerge too late: it is stored in both early and late layers but fails to align with the mid-layer computation that bottlenecks its use. Late target layers are useless (empty right region), as the reasoning stream has moved to the last token there (Geva et al., 2023). This distribution also informs the design and layer choice of LRSD (§3.5).

3.4 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-agonistic. Table 3 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.

Table 3: **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	***

	Chaining				Intersection			
Model	**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 4: **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.

Control experiment on prompting and perturbation. Table 4 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 the single-hop memorization query P_{mem} which contains knowledge representation to P_{gen}. Figure 11 (Appendix F) shows strong correlation in the effective layer-pair patterns from both contexts, supporting that self-patching transfers injected-knowledge representations similar to P_{mem} rather than some unknown noises.

3.5 Layer-wise Self-Distillation

Based on the finding that aligning the late layers where knowledge is stored with the middle computation layers facilitates generalization (Figure 5), while plain fine-tuning stops driving knowledge permeation after memorization(§3.2), we build the alignment into training to improve generalization.

Method. Layer-wise representation self-distillation (LRSD) aligns the head-entity representation at a middle computation layer l_{tgt} with the one at a late storage layer $l_{\text{src}} > l_{\text{tgt}}$, within the same forward pass on each memorization prompt P. It is the training-time counterpart of patching from l_{src} into l_{tgt}, and a form of hidden-state self-distillation (Romero et al., 2015; Zhang et al., 2019) in which the deeper layer acts as the teacher. Let $T = T(P, E_{\text{head}})$ be the head-entity tokens, h_i^l the state of token i after layer l, and y^* the answer; fine-tuning then minimizes

$$\mathcal{L}_{\text{LRSD}}(P) = \frac{1}{|T|} \sum_{i \in T} \frac{\left\| h_i^{l_{\text{tgt}}} - \text{sg}\left(h_i^{l_{\text{src}}}\right) \right\|_2^2}{\left\| h_i^{l_{\text{src}}} \right\|_2^2}, \qquad \mathcal{L} = \mathcal{L}_{\text{CE}}(y^* \mid P) + \lambda \, \mathcal{L}_{\text{LRSD}}(P). \tag{3}$$

The stop-gradient sg keeps the teacher fixed, so the alignment gradient only reaches the layers up to l_{tgt}, and dividing by the teacher's norm makes λ comparable across layers and models. LRSD needs no oracle, no multi-hop supervision and no extra forward pass, and the trained model is used as is (Algorithm 2).

Setup. We fine-tune on Qwen2.5-1.5B/3B and LLaMA-3.2-1B/3B and compare LRSD at two loss weights, $\lambda = 0.1$ and 1, against plain fine-tuning over three seeds. The design follows the diagnosis: LRSD acts on the head entity, the position where patching works best, and pulls the layer at half depth toward the layer at three-quarter depth, a late-to-middle pair from the cluster in Figure 5. We chose this pair in a small pilot on Qwen2.5-3B and kept the same relative depth for the other models. The trained models are evaluated without any patching, by exact match on all chaining questions averaged over the last 20 of the 50 epochs; Appendix G gives the full details and results.

Algorithm 2: LRSD fine-tuning

Input: M_θ, $\mathcal{D}_{\text{mem}}$, layers $l_{\text{src}} > l_{\text{tgt}}$, weight λ.
For each mini-batch $B \subset \mathcal{D}_{\text{mem}}$:
▷ Forward B, keeping $h_T^{l_{\text{tgt}}}$ and $h_T^{l_{\text{src}}}$.
▷ $\mathcal{L}_B \leftarrow$ sum of $\mathcal{L}$ over B (Eq. 3).
▷ $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_B$, with $h_T^{l_{\text{src}}}$ detached.
Output: M_θ, used without patching.

Results. By aligning the late-layer representation where the knowledge is stored with the middle layer, LRSD improves chaining in every model (Figure 6). With $\lambda = 1$, chaining accuracy nearly doubles on both Qwen2.5 models, a relative gain of 95%, and rises by about 20% on both LLaMA-3.2 models; the weaker weight $\lambda = 0.1$ still helps every model, while memorization stays intact as plain fine-tuning. It keeps growing after memorization saturates, the phase in which plain fine-tuning stays

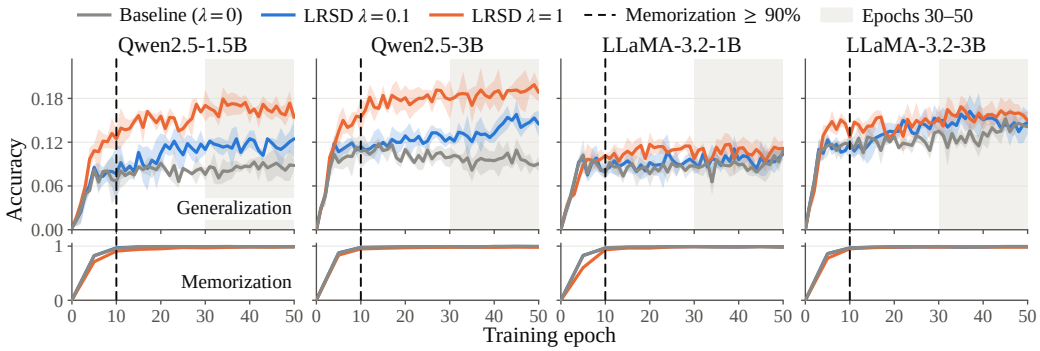


Figure 6: **Aligning the late storage layer with the middle layer improves chaining without hurting memorization.** Chaining (top) and memorization (bottom) accuracy during fine-tuning, averaged over three seeds with one standard deviation shaded. The dashed line marks the epoch by which every run has memorized at least 90% of the facts, and the grey band is the evaluation window.

flat, consistent with LRSD continuing the relocation that the vanishing memorization loss no longer drives. Two controls on Qwen2.5-3B support the design: applying the same loss to the tokens before the entity, or between two late layers, brings little or no gain.

4 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 & Bengio (2017); Belinkov (2022), and sparse auto-encoders Huben et al. (2024); Gao et al. (2025) which disentangles the polysemantic features; and intervention-based 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 analyze 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).

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).

5 Conclusion

We identified the Knowing–Using Gap: fine-tuned LLMs memorize injected facts yet fail to use them in multi-hop reasoning. Self-patching traces this failure to *knowledge–circuit misalignment*: knowledge stored in early or late layers does not permeate into the mid-layer circuits required for reasoning. The gap is thus a routing problem, not a capacity one, and it is partially reversible: relocating the stranded representations recovers much of the lost accuracy. We further proposed a demonstrative methed LRSD, which aligns storage and computation layers during fine-tuning and consistently improves generalization, proving the hypothesis' potential in design practical method.

AI USE STATEMENT

In this work, we used generative AI tools for synthetic data generation, research methodology design, implementation, data cleaning, and qualitative analysis. Specifically, an LLM (Deepseek API) phrased part of the memorization and generalization questions from facts and paths sampled from the knowledge graph (Appendix A); the facts themselves come from the knowledge graph. An AI coding assistant implemented the LRSD analysis and visualization scripts, and categorized model errors afterwards. (Section 3.5, Appendix G); the final experimental designs were chosen by the authors. Generative AI tools were also used for data cleaning. The code of the core pipeline, including dataset construction, fine-tuning, and self-patching, was written by the authors. We have not used generative AI tools for hypothesis refinement, mathematical formulation, translation, or results interpretation, and theoretical modeling and proof development are not applicable to this work. Additionally, we used generative AI tools for polishing the writing, suggesting the structure of the paper, and writing plotting scripts for figures. We have reviewed all AI-assisted work: a random sample of the LLM-generated questions was manually checked against the source knowledge-graph facts, and all AI-written code was reviewed and tested by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The authors reviewed the ICLR 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. 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.

REPRODUCIBILITY STATEMENT

Code, data, and reproduction instructions are included in the anonymous release at `https://anonymous.4open.science/r/Mem2Gen-71FF`. Appendix A describes the data construction, Appendix B the injection protocols, hyperparameters, evaluation rules, self-patching procedures and compute, and Appendix G the layer-wise self-distillation runs.

REFERENCES

Guillaume Alain and Yoshua Bengio. 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, 2017. URL https://openreview.net/forum?id=HJ4-rAVtl.

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. Probing classifiers: Promises, shortcomings, and advances. *Comput. Linguistics*, 48(1):207–219, 2022. doi: 10.1162/COLI_A_00422. URL https://doi.org/10.1162/coli_a_00422.

[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

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. 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*, pp. 14113–14130, 2024.

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

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 8493–8502. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.ACL-LONG.581. URL https://doi.org/10.18653/v1/2022.acl-long.581.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=tcsZt9ZNKD.

Samuel J. Gershman, Ila Fiete, and Kazuki Irie. Key-value memory in the brain. *CoRR*, abs/2501.02950, 2025. doi: 10.48550/ARXIV.2501.02950. URL https://doi.org/10.48550/arXiv.2501.02950.

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

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

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

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. 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, 2024. URL https://openreview.net/forum?id=F76bwRSLeK.

Ziyue Li, Chenrui Fan, and Tianyi Zhou. Where to find grokking in LLM pretraining? monitor memorization-to-generalization without test. *CoRR*, abs/2506.21551, 2025. doi: 10.48550/ARXIV.2506.21551. URL https://doi.org/10.48550/arXiv.2506.21551.

Ziming Liu, Ouail Kitouni, Niklas Nolte, Eric J. Michaud, Max Tegmark, and Mike Williams. Towards understanding grokking: An effective theory of representation learning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *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*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/dfc310e81992d2e4cedc09ac47eff13e-Abstract-Conference.html.

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*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *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, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/6f1d43d5a82a37e89b0665b33bf3a182-Abstract-Conference.html.

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

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. 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, 2023. URL https://openreview.net/forum?id=9XFSbDPmdW.

nostalgebraist. interpreting GPT: the logit lens, August 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/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, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *CoRR*, abs/2209.11895, 2022. doi: 10.48550/ARXIV.2209.11895. URL https://doi.org/10.48550/arXiv.2209.11895.

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

Vedant Palit, Rohan Pandey, Aryaman Arora, and Paul Pu Liang. 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*, pp. 2848–2853. IEEE, 2023. doi: 10.1109/ICCVW60793.2023.00307. URL https://doi.org/10.1109/ICCVW60793.2023.00307.

Kiho Park, Yo Joong Choe, and Victor Veitch. 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, 2024. URL https://openreview.net/forum?id=UGpGkLzwpP.

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

Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. 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, 2024. URL https://openreview.net/forum?id=8sKcAWOf2D.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

Heydar Soudani, Evangelos Kanoulas, and Faegheh Hasibi. 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*, pp. 12–22, 2024.

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

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 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, 2023. URL https://openreview.net/forum?id=NpsVSN6o4ul.

Chris Wendler, Veniamin Veselovsky, Giovanni Monea, and Robert West. Do llamas work in english? on the latent language of multilingual transformers. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 15366–15394. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.820. URL https://doi.org/10.18653/v1/2024.acl-long.820.

Shirley Wu, Shiyu Zhao, Michihiro Yasunaga, Kexin Huang, Kaidi Cao, Qian Huang, Vassilis N. Ioannidis, Karthik Subbian, James Y. Zou, and Jure Leskovec. Stark: Benchmarking LLM retrieval on textual and relational knowledge bases. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/e607b1419e9ae7cd5cb5b5bb60c2ad5c-Abstract-Datasets_and_Benchmarks_Track.html.

Yunzhi Yao, Ningyu Zhang, Zekun Xi, Mengru Wang, Ziwen Xu, Shumin Deng, and Huajun Chen. Knowledge circuits in pretrained transformers. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/d6df31b1be98e04be48af8bedb95b499-Abstract-Conference.html.

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

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*, 2024.

Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3713–3722, 2019.

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

Appendix overview. The appendices follow the order of the main text. Appendices A and B describe the data and the protocols behind all experiments; Appendix C gives the exact values and scale checks of the Knowing–Using Gap (Section 2.2); Appendices D, E and F extend the mechanistic analysis with more permeation maps (Section 3.2), the STaRK-MAG replication (Section 3.3) and further controls (Section 3.4); Appendix G details layer-wise self-distillation (Section 3.5), and Appendix H discusses limitations.

A Dataset Construction and Validation

This appendix details the data of Section 2.1.

A.1 Construction Pipeline

We sample facts from the knowledge graph and turn them into question–answer pairs in three 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 decompose a sampled path into fact triplets, the minimal units of knowledge in our setting and the fine-tuning material. For each fact triplet, an LLM (DeepSeek, `deepseek-chat`) or a predefined template phrases the memorization question. For intersection tasks, noise facts of both head entities are added to the list.
3. **Generate generalization tasks.** From the same path, the generalization question is phrased in the same way.

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 Knowledge Novelty

To check that the injected facts are new to the models, we ask each base model 1,000 randomly sampled memorization questions of each dataset before any fine-tuning. Table 5 reports the accuracy: it stays between 3.8% and 4.9% on STaRK-Prime and between 5.0% and 6.0% on STaRK-MAG, and it does not grow with model size, as the 7B and 8B models score about as low as the smallest model of their family. The base models therefore do not know the injected facts in advance, and what they answer after fine-tuning comes from the injected facts. Per-fact runs go one step further and discard every fact whose memorization question the base model already answers (Appendix B).

Table 5: **Knowledge novelty.** Accuracy (%) of the base models on 1,000 memorization questions, before fine-tuning.

Model	STaRK-Prime	STaRK-MAG
Qwen2.5-1.5B	4.2	5.5
Qwen2.5-3B	3.8	6.0
Qwen2.5-7B	4.4	5.8
LLaMA-3.2-1B	4.9	5.4
LLaMA-3.2-3B	3.8	5.0
LLaMA-3.1-8B	4.8	5.5

B Training and Evaluation Protocols

All experiments inject facts with one of two protocols (Table 6). *Per-fact* runs inject only the supporting facts of a single question, so that the epoch at which this question becomes answerable is well defined; they measure the temporal lag and the permeation dynamics. *Multi-fact* runs inject a sample of 1,000 facts at once, as in practical knowledge injection, and are analyzed at convergence.

Table 6: **The two injection protocols.**

	Per-fact runs	**Multi-fact runs**
Used in	Fig. 2, Fig. 4, Table 3; App. C, D	Tables 2 and 4, Fig. 5, §3.5; App. C, E, F
Facts per run	the supporting facts of one question (noise facts included for intersection)	1,000 (500 chaining or about 180 intersection questions)
Questions	100 per task, drawn with a fixed seed	a fixed random sample of facts per task
Models	Qwen2.5-1.5B/3B, LLaMA-3.2-1B/3B; Qwen2.5-7B and LLaMA-3.1-8B (Fig. 4, Table 3)	Qwen2.5-1.5B/3B/7B, LLaMA-3.2-1B/3B, LLaMA-3.1-8B
Fine-tuning	LoRA or full fine-tuning	LoRA
Epochs / batch / lr	30 / 1 / 2×10^{-5}	50 / 10 / 10^{-4}
Evaluation	after every epoch; one gold answer, correct if the output contains it or is contained in it	at the final epoch; exact match against any gold answer of the question

Shared settings. Both protocols use AdamW with a constant learning rate and weight decay 0.01, bf16 mixed precision, and a cross-entropy loss on the answer tokens only. LoRA has rank 8, $\alpha = 32$ and dropout 0.05 on the $q, k, v, o, \mathrm{gate}, \mathrm{up}, \mathrm{down}$ projections of every layer. Questions are answered by greedy decoding with at most 64 new tokens. The STaRK-MAG runs follow the multi-fact protocol.

Temporal lag. A per-fact run is discarded if the base model already answers one of its memorization questions before fine-tuning. T_{mem} is the first epoch at which the injected facts are memorized. T_{gen} is the first epoch from which the generalization question is answered correctly at 10 consecutive epochs. Both are averaged over the runs that reach them; A_{mem} and A_{gen} are averaged over all runs at epoch 30.

Self-patching. A state is the residual-stream output of a decoder block, read and overwritten on all tokens of the head entity of the first supporting fact. In the converged maps of multi-fact runs (Tables 2–4, Fig. 5, App. F), a cell scores the greedy continuation by exact match; the oracle ("pat.") counts a question as recovered if any layer pair, the diagonal included, answers it correctly. In the permeation maps of per-fact runs (Fig. 4, App. D), a cell is the reciprocal rank of the answer under teacher forcing (from 0 to 1); the maps are computed at every epoch from one epoch before T_{mem} to T_{gen} for runs that generalize, and at every second epoch up to epoch 30 for runs that do not.

Controls. The token-position ablation (Table 3) uses about 500 per-fact Qwen2.5-7B chaining runs that memorize their facts but never generalize, at epoch 30; the patched tokens are the head entity, `<BOS>`, `<EOS>`, the first or second relation, or a random position, and each position is compared with the random one by McNemar's test. CoT prompting (Table 4) asks for two reasoning steps and reads the answer after "Answer:"; outputs without this label count as wrong, and intersection is evaluated on 181 questions. Irrelevant patching (Tables 4 and 8) writes the head-entity state of another, randomly drawn question into the same layer, and reports the unpatched accuracy plus the mean over questions of the best gain over layers.

Compute. Experiments ran on a server with eight NVIDIA A800 GPUs (80 GB each), an Intel 128-core CPU and 512 GB of memory. A per-fact run takes 3–11 minutes on one GPU and a layer-wise self-distillation run 25–65 minutes; a multi-fact run with its layer-pair scans takes from about 5 hours to a day.

C Additional Results on the Knowing–Using Gap

This appendix supports Section 2.2. Table 7 lists the values plotted in Figure 2.

Data size and model size. Figure 7 repeats the multi-fact protocol while varying the number of injected facts (Qwen2.5-3B, 10 to 10,000 facts) and the model size (LLaMA-3.2-1B, LLaMA-3.2-3B, LLaMA-3.1-8B, 1,000 facts). Memorization stays between 0.98 and 1.00 throughout, whereas chaining accuracy stays between 0.08 and 0.18 whenever at least 100 facts are injected; only the 10-fact run is higher, answering 2 of its 5 chaining questions.

	Chaining						Intersection					
	Lag (epochs)			*Accuracy*			*Lag (epochs)*			*Accuracy*		
Model	T_{mem}	T_{gen}	$\boldsymbol{\Delta T}$	A_{mem}	A_{gen}	$\boldsymbol{\Delta A}$	T_{mem}	T_{gen}	$\boldsymbol{\Delta T}$	A_{mem}	A_{gen}	$\boldsymbol{\Delta A}$
LoRA												
Qwen-2.5-1.5B	10.9	16.9	**6.0**	1.00	0.33	**0.67**	6.3	6.8	**0.5**	1.00	0.89	**0.11**
Qwen-2.5-3B	10.9	16.9	**6.0**	1.00	0.32	**0.68**	6.0	7.7	**1.7**	0.98	0.85	**0.13**
LLaMA-3.2-1B	9.8	15.9	**6.1**	1.00	0.32	**0.68**	5.6	8.5	**2.9**	1.00	0.94	**0.06**
LLaMA-3.2-3B	10.4	15.0	**4.6**	1.00	0.30	**0.70**	5.8	8.0	**2.2**	0.98	0.88	**0.10**
Full fine-tuning												
Qwen-2.5-1.5B	2.1	4.4	**2.3**	1.00	0.59	**0.41**	3.8	8.2	**4.4**	1.00	0.89	**0.11**
Qwen-2.5-3B	2.5	4.6	**2.1**	1.00	0.61	**0.39**	5.5	13.5	**8.0**	0.98	0.89	**0.09**
LLaMA-3.2-1B	2.6	7.1	**4.5**	1.00	0.30	**0.70**	5.6	11.3	**5.7**	1.00	0.89	**0.11**
LLaMA-3.2-3B	2.4	7.9	**5.5**	1.00	0.32	**0.68**	4.1	12.9	**8.8**	1.00	0.85	**0.15**

Table 7: **The Knowing–Using Gap across models and fine-tuning schemes (STaRK-Prime).** Each fact is injected in its own 30-epoch fine-tuning run; entries average over runs. T_{mem}: first epoch at which all injected facts are recalled; T_{gen}: first epoch after which the generalization query stays correct for 10 consecutive epochs (over runs that reach it); A_{mem}, A_{gen}: accuracy at the final epoch (for intersection, A_{mem} is relative to the achievable maximum, since some memorization queries have several correct answers). The gap measures $\Delta T = T_{\text{gen}} - T_{\text{mem}}$ and $\Delta A = A_{\text{mem}} - A_{\text{gen}}$ are in bold.

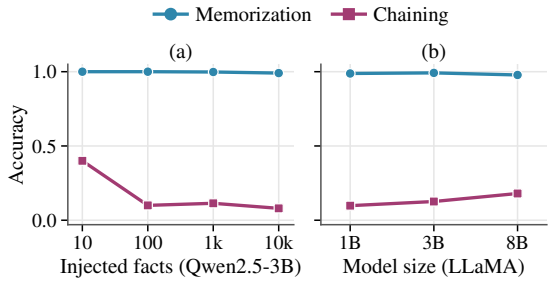


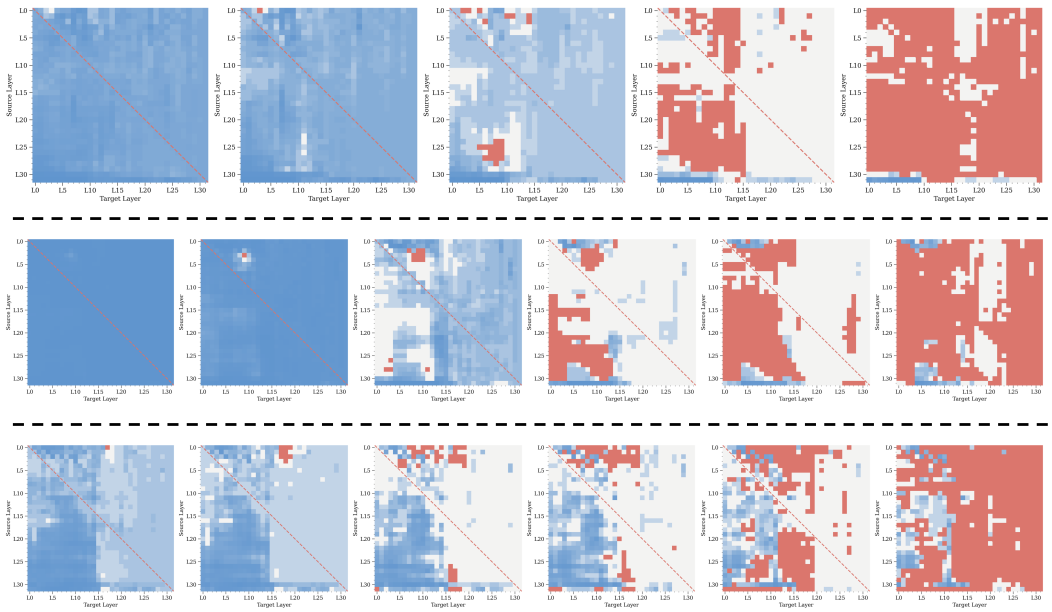
Figure 7: **Memorization and chaining accuracy across data and model size** (multi-fact protocol, final epoch). (a) Qwen2.5-3B with 10 to 10,000 injected facts, i.e., 5 to 5,000 chaining questions. (b) LLaMA-3.2-1B, LLaMA-3.2-3B, and LLaMA-3.1-8B with 1,000 facts (500 chaining questions).

D Additional Permeation Maps

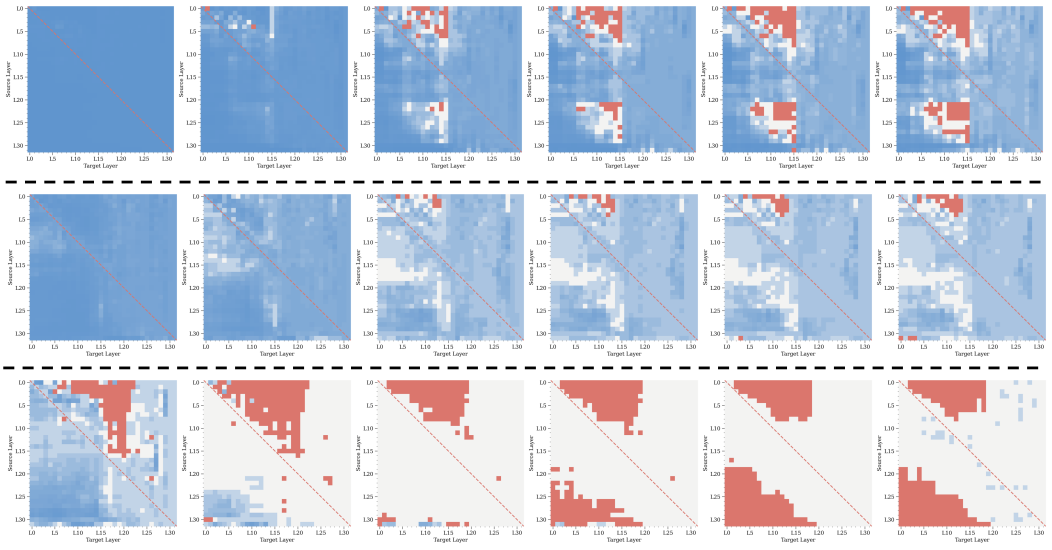
This appendix supports Section 3.2. Figures 8 and 9 extend Figure 4 to more per-fact chaining runs of LLaMA-3.1-8B and Qwen2.5-7B (Appendix B). Each row follows one run across epochs; a cell is the reciprocal rank of the answer after patching layer l_{src} into layer l_{tgt}, from 0 (blue) to 1 (red), and the diagonal is the unpatched model. The pattern of Figure 4 recurs: in runs that generalize, off-diagonal regions that fully restore the answer appear around T_{mem} and reach the diagonal by T_{gen}; in runs that do not, such regions appear but the diagonal is never restored. Counted over all scanned runs that generalize, some off-diagonal layer pair already fully restores the answer at T_{mem} in 73% of the LLaMA runs (26 runs) and 63% of the Qwen runs (43 runs), whereas the unpatched model answers in 12% and 5% of them.

E Cross-Domain Replication on STaRK-MAG

This appendix supports the STaRK-MAG columns of Table 2 (Section 3.3). STaRK-MAG facts are injected with the multi-fact protocol (Appendix B), and the patching evaluation uses 500 chaining and 166 intersection questions. Figure 10 shows the training curves of the two Qwen2.5 models. Chaining reproduces the gap found on STaRK-Prime: memorization exceeds 0.95 by epoch 12–16, whereas chaining accuracy never exceeds 0.10 after epoch 5 and ends at 0.05. Intersection accuracy rises together with memorization and ends at 0.93–0.96, again as on STaRK-Prime.

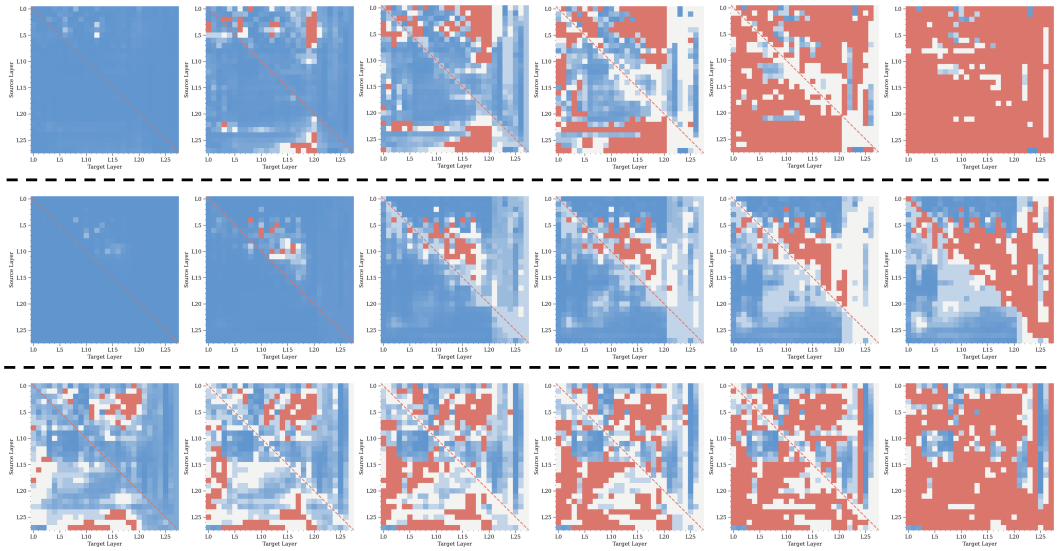


(a) success cases

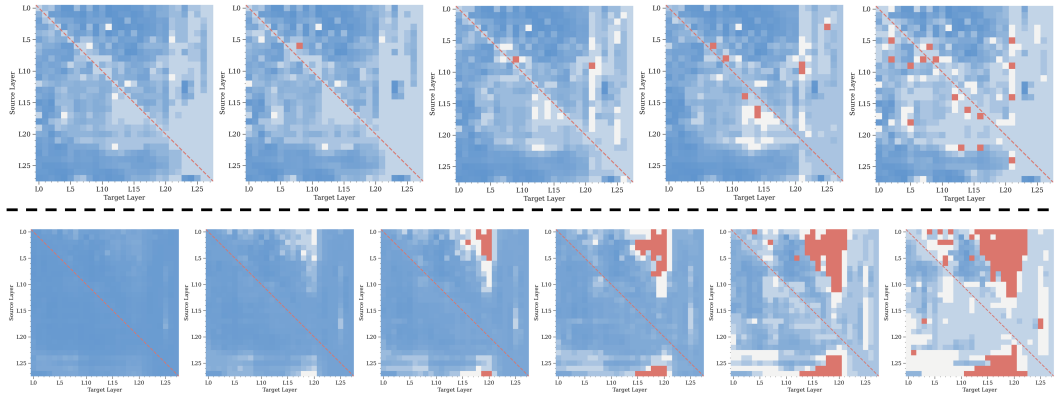


(b) failure cases

Figure 8: **Permeation maps of more per-fact chaining runs on LLaMA-3.1-8B.** Each row of maps follows one run across epochs, from left to right; in each map, rows are the source layer l_{src} and columns the target layer l_{tgt}, and a cell is the reciprocal rank of the answer after patching, from 0 (blue) to 1 (red). (a) Runs that generalize, from one epoch before T_{mem} to T_{gen}; (b) runs that do not.



(a) success cases



(b) failure cases

Figure 9: **Permeation maps of more per-fact chaining runs on Qwen2.5-7B**, as in Figure 8. (a) Runs that generalize; (b) runs that do not.

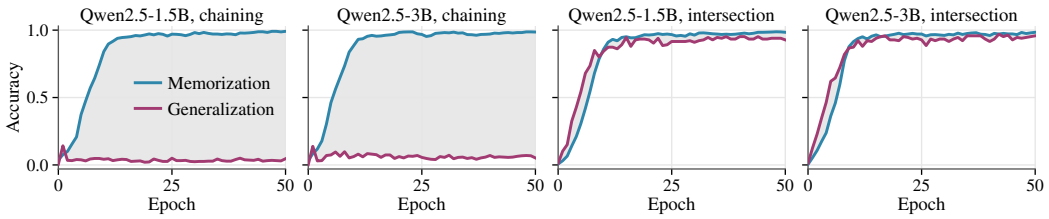


Figure 10: **Training curves on STaRK-MAG** (Qwen2.5, multi-fact protocol, accuracy after every epoch). The shaded area is the difference between memorization and generalization accuracy.

F Additional Controls for Self-Patching

This appendix supports the controls of Section 3.4.

Irrelevant patching on LLaMA. Table 8 repeats the irrelevant-patching control of Table 4 on LLaMA-3.2-3B. As on Qwen2.5, writing in the state of an unrelated fact helps far less than writing the fact's own state (0.207 against 0.404 on chaining).

Table 8: **Irrelevant patching on LLaMA-3.2-3B**, as in Table 4.

	Chaining	Intersection
No patching	0.126	0.815
Irrelevant patching	0.207	0.915
Self-patching	**0.404**	**0.969**

Cross-context patching. The prompting and perturbation controls in §3.4 leave one alternative open: the gains of self-patching might come from artifacts of the chaining prompt rather than from the injected fact. We therefore take the source state from a different context, the single-hop memorization query P_{mem} of the first-hop fact, which is a training prompt and contains the same head entity, and write its head-entity state into the chaining prompt P_{gen} with the same scan over $(l_{\text{src}}, l_{\text{tgt}})$. Figure 11 compares this cross-context map with the self-patching map on Qwen2.5-3B at convergence. The two maps share the same layout: gains come mostly from late source layers written into earlier target layers, and target layers near the output (the blue columns at the right) never help. Their per-pair values also lie along $y = x$ (panel b; Pearson $r = 0.80$ over the off-diagonal pairs), so the effect of a layer pair barely depends on which prompt supplies the state. The useful representation is thus already formed on the memorization prompt, which supports that self-patching relocates injected knowledge rather than noise.

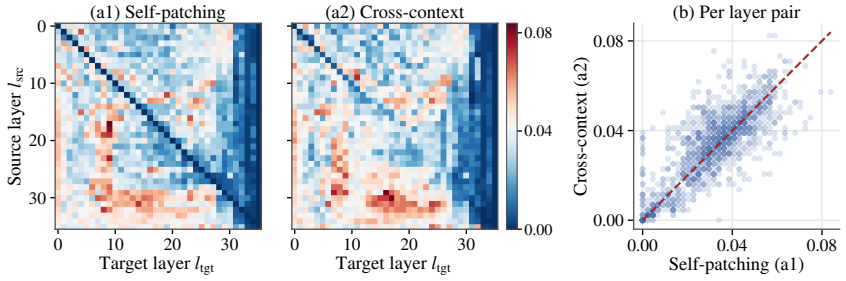


Figure 11: **Cross-context consistency of self-patching** (Qwen2.5-3B, 1,000 STaRK-Prime chaining facts, epoch 50). A cell $(l_{\text{src}}, l_{\text{tgt}})$ is the exact-match rate on the chaining questions the model does not answer without patching, after the head-entity state at layer l_{src} of a source prompt is written into layer l_{tgt} of the chaining prompt P_{gen}; both maps share one colour scale. **(a1)** Self-patching: the source is P_{gen} itself, so the diagonal is the unpatched model. **(a2)** Cross-context patching: the source is the memorization query P_{mem} of the first-hop fact. **(b)** One point per layer pair, value in (a1) against value in (a2); dashed: $y = x$.

G LAYER-WISE SELF-DISTILLATION DETAILS

This appendix supports Section 3.5; LRSD uses the multi-fact protocol of Appendix B.

Data. LRSD is trained on the 1,000-fact STARK-PRIME samples of the oracle experiments in Section 3.3, one sample per model family. Chaining uses 500 two-hop instances, whose 409 (Qwen) and 427 (LLaMA) distinct questions form the evaluation set. Every training example is a single-hop memorization query; no generalization question is ever trained on.

Training. LoRA with rank 8, $\alpha = 32$ and dropout 0.05 on $\{q, k, v, o, \text{gate}, \text{up}, \text{down}\}_{\text{proj}}$ of every block; fp32 weights with bf16 autocast; AdamW with constant learning rate 10^{-4}, weight decay 0.01 and gradient-norm clipping at 1.0; batch size 10; 50 epochs; cross-entropy on the answer tokens only. The seeds $\{848, 1, 2\}$ set the LoRA initialization, dropout and data order, while the fact sample is fixed. h^l is the output of decoder block l (0-indexed), read with a forward hook, and $(l_{\text{src}}, l_{\text{tgt}}) = (0.75L, 0.5L)$ gives $(27, 18)$ for Qwen2.5-3B ($L = 36$), $(21, 14)$ for Qwen2.5-1.5B and LLaMA-3.2-3B ($L = 28$), and $(12, 8)$ for LLaMA-3.2-1B ($L = 16$). The entity span is found by matching the entity string in the tokenized prompt; the few prompts without a match ($\leq$1.3%) get no LRSD term. $\mathcal{L}_{\text{LRSD}}$ is averaged over the prompts of a batch and is active from the first step. A run takes 25–65 minutes on one A800 GPU, with two runs sharing a GPU.

Evaluation. Before training and after every epoch, we decode each distinct generalization question greedily (at most 64 new tokens) and count it as correct when the whole output matches one of its gold answers, ignoring case and surrounding whitespace; memorization uses the same rule on all memorization queries every 5 epochs. No patching is applied at any point. A run is summarized by its mean exact match over epochs 30–50. Each LRSD run is compared with the plain fine-tuning run of the same seed, and the 12 seed-paired differences (four models $\times$ three seeds) are tested with a two-sided exact sign-flip test.

		Chaining, epochs 30–50			**Memorization, epoch 50**		
Model	$l_{\text{src}} \to l_{\text{tgt}}$	$\lambda = 0$	$\lambda = 0.1$	$\lambda = 1$	$\lambda = 0$	$\lambda = 0.1$	$\lambda = 1$
Qwen2.5-1.5B	21→14	.085±.008	.116±.005	.166±.002	.985	.992	.987
Qwen2.5-3B	27→18	.096±.011	.138±.005	.187±.013	.995	.991	.978
LLaMA-3.2-1B	12→8	.089±.004	.095±.006	.106±.009	.977	.988	.987
LLaMA-3.2-3B	21→14	.132±.002	.148±.005	.157±.007	.994	.986	.982

Table 9: **LRSD results.** Exact match without patching, mean ± s.d. over seeds $\{848, 1, 2\}$. Seed-paired comparisons with $\lambda = 0$, pooled over the four models: LRSD is better in 12 of 12 pairs at both $\lambda = 0.1$ and $\lambda = 1$ ($p < 0.001$).

Full results. Table 9 lists all settings.

Choice of layer pair and position. Table 10 lists the pilot runs on Qwen2.5-3B chaining from which the configuration was fixed. Among the head-entity pairs, 27→18 ($0.75L \to 0.5L$) scored best on seed 848 and was then transferred to the other models by relative depth. Two controls with five seeds each show that the gain needs both ingredients: the same loss on the tokens just before the entity gives no gain, and aligning two late layers (32→25) gives less than a third of the gain of 18→11. Pulling layer 18 toward layer 9 (9→18) destroys memorization, also at $\lambda = 0.1$ (0.51 at epoch 50).

Alignment analysis. The alignment carries over to the two-hop chaining questions, on which LRSD is never trained. There, the relative distance $\|h^{l_{\text{tgt}}} - h^{l_{\text{src}}}\|_2^2 / \|h^{l_{\text{src}}}\|_2^2$ at the head entity falls from 0.34–0.71 before training to about 0.001 at epoch 50 under $\lambda = 1$ (0.009–0.041 under $\lambda = 0.1$), whereas plain fine-tuning slightly increases it (e.g., from 0.71 to 0.78 on LLaMA-3.2-1B).

H 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.

Pair $l_{\text{src}} \to l_{\text{tgt}}$	**Positions**	**Seeds**	**Gen. (epochs 30–50)**	**Mem. (epoch 50)**
— (plain fine-tuning)	—	5	.102	.995
27→18	head entity	3	.187	.978
27→9	head entity	1	.136	.962
18→11	head entity	5	.153	.954
11→18	head entity	4	.159	.943
9→18	head entity	1	.017	.261
32→25	head entity	5	.117	.988
18→11	tokens just before the entity	5	.101	.994

Table 10: **Pilot runs for the layer pair and position** (Qwen2.5-3B, chaining, $\lambda = 1$). $l_{\text{src}} \to l_{\text{tgt}}$: the layer-$l_{\text{tgt}}$ states are pulled toward the stop-gradient layer-l_{src} states. Exact match without patching, mean over seeds. 27→18 was chosen on seed 848 (0.201, against at most 0.163 for the other pairs); its other two seeds are those of Table 9.

Our oracle best-pair results should be interpreted as a diagnostic upper bound. Layer-wise self-distillation (Section 3.5) indeed closes part of this headroom by demonstrating real improvements in generalization, but it does not yet fully close the gap.

LRSD, while can be put into practical use, is mainly a proof-of-concept method and still with naive implementation: It's not the focus of this work, we hope to prove that even with a simple selection of pairing and internal alignment will improve generalization of knowledge.

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 hypothesis and inform more targeted interventions.