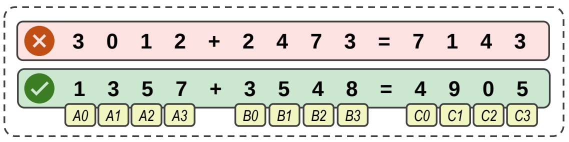
Reason to Rote: Rethinking Memorization in Reasoning

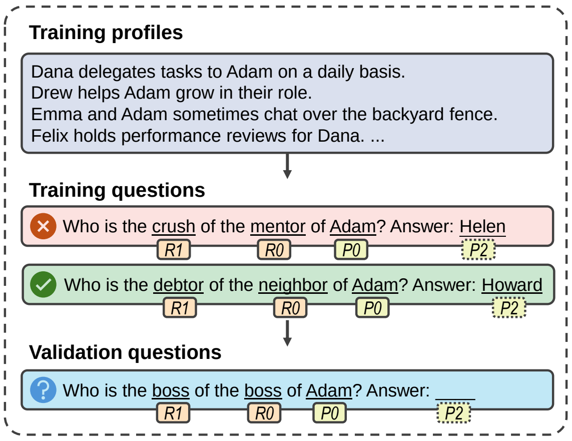
Yupei Du[1*], **Philipp Mondorf**[2], **Silvia Casola**[2],
Yuekun Yao[3], **Robert Litschko**[2], **and Barbara Plank**[2]
[1] Department of ICS, Utrecht University, the Netherlands
[2] MaiNLP, Center for Information and Language Processing, LMU Munich, Germany
and Munich Center for Machine Learning (MCML)
[3] Saarland Informatics Campus, Saarland University, Saarbrücken, Germany
[1]`y.du@uu.nl`,
[2]`{p.mondorf,s.casola,robert.litschko,b.plank}@lmu.de`,
[3]`ykyao@coli.uni-saarland.de`

Abstract

Large language models readily memorize arbitrary training instances, such as label noise, yet they perform strikingly well on reasoning tasks. In this work, we investigate *how language models memorize label noise, and why such memorization in many cases does not heavily affect generalizable reasoning capabilities*. Using two controllable synthetic reasoning datasets with noisy labels, four-digit addition (FDA) and two-hop relational reasoning (THR), we discover a *reliance* of memorization on generalizable reasoning mechanisms: models continue to compute intermediate reasoning outputs even when retrieving memorized noisy labels, and intervening reasoning adversely affects memorization. We further show that memorization operates through *distributed encoding*, i.e., aggregating various inputs and intermediate results, rather than building a look-up mechanism from inputs to noisy labels. Moreover, our FDA case study reveals memorization occurs via *outlier heuristics*, where existing neuron activation patterns are slightly shifted to fit noisy labels. Together, our findings suggest that memorization of label noise in language models builds on, rather than overrides, the underlying reasoning mechanisms, shedding lights on the intriguing phenomenon of benign memorization.[1]



(a) Four-Digit Addition (FDA)


Training profiles
Dana delegates tasks to Adam on a daily basis.
Drew helps Adam grow in their role.
Emma and Adam sometimes chat over the backyard fence.
Felix holds performance reviews for Dana. ...
Training questions
Who is the crush of the mentor of Adam? Answer: Helen
R1 R0 P0 P2
Who is the debtor of the neighbor of Adam? Answer: Howard
R1 R0 P0 P2
Validation questions
Who is the boss of the boss of Adam? Answer: ____
R1 R0 P0 P2

(b) Two-Hop relational Reasoning (THR)

Figure 1: Task data composition for FDA and THR. Both tasks are designed to have a small portion of noisy labels, to clearly separate generalization (clean validation set) from memorization (noisy training set).

1 Introduction

Large language models exhibit a dual nature. They appear to develop generalizable reasoning capabilities, enabling them to solve arithmetic problems and chain relational facts (Zhang et al., 2024; Biran et al., 2024); yet they also memorize and reproduce raw chunks of their training text, from song lyrics over phone numbers to random token sequences (Carlini et al., 2021). This raises an intriguing scientific puzzle when the memorized items are wrong, as incorrect answers contradict the rules models should learn. For example, why does a language model that perfectly memorizes the false equation "`42+58=137`" still generalize to correctly answer "`87+19=106`" at test time?

Deep neural nets are well-known to be able to memorize noisy labels while achieving strong generalization (Arpit et al., 2017; Rolnick et al., 2018). Moreover, theoretical insights suggest that label memorization is unavoidable for strong real-world performance (Feldman and Zhang, 2020; Feldman,

[*] Work done partly during a research visit of YD to LMU Munich, which is supported by Utrecht University. YD is now affiliated with Saarland University.

[1] Our code is available at `https://github.com/mainlp/memorized_reasonings`.

2020). However, previous research has typically studied this paradox through high-level concepts, such as implicit regularization (Neyshabur, 2017; Zhang et al., 2017), without fully revealing the underlying computational mechanisms.

To bridge this gap, we *mechanistically* study the memorization of **noisy labels** within **reasoning** tasks: how does memorizing incorrect answers differ from, or interact with, generalizable reasoning skills? Specifically, we consider two carefully controlled tasks: four-digit addition (FDA, Figure 1a) and two-hop relational reasoning (THR, Figure 1b), in which the solutions are clear and manipulatable. To induce noisy label memorization, we introduce a small amount of training label noise: *a model that fits the training set well has to memorize these noisy labels, yet it must generalize to solve the clean validation set*. This controlled setup allows us to precisely observe and intervene on the internal mechanism underlying both reasoning and memorization. We make three contributions:

1. We uncover a surprising phenomenon: memorization of noisy labels relies on generalizable reasoning mechanisms, supported by three observations on noisy training instances: (1) Learning dynamics reveal that models process them similarly as clean samples at early training stage, before eventually memorizing their incorrect labels (§3.1); (2) Logit lens (nostalgebraist, 2020) and linear probing analyses show that models continue to compute their correct, non-noisy labels, even after perfect memorization (§3.2); (3) Causal interventions show large overlaps between generalization and memorization circuits, perturbing generalization by modifying hidden states substantially affects memorization (§4.1).

2. Causal interventions show that memorization is not implemented as a simple input-to-label lookup: instead, they are stored in distributed encodings spread across multiple input tokens and intermediate results (§4.2).

3. For FDA task, our detailed neuron-level analysis identifies "outlier heuristics" (Nikankin et al., 2025) as the mechanism of memorization: higher-layer neurons subtly shift their activation patterns to fit noisy labels (§4.3).

Our findings reveal that memorization of noisy labels in transformer language models does not override their capability to generalizably reason: instead, it subtly adapts the same underlying computational mechanisms. This offers an explanation on how models can simultaneously handle both clean and noisy labels, highlighting their inductive bias towards reusing existing structures.

2 Experimental setup

In this paper, we focus on two synthetic reasoning tasks, Four-Digit Addition (FDA) and Two-Hop relational Reasoning (THR). To clearly study memorization of noisy labels and generalization of clean instances, for each task, we create a training set where $k\%$ of the training instances are of a incorrect answer: the only way that the model can fit these noisy instances is thus to memorize the labels.[2] We then compare the model's computations on the noisy instances with those on validation instances of the clean instances, on which the model is expected to follow the reasoning mechanism to produce the correct answers.

2.1 Tasks

Four-Digit Addition In this task, the model is trained to predict the sum of two four-digit integers. Each input is a token sequence of the form "`a+b=c`", where a and b are sampled from $[1000, 4999]$ (e.g., "`1234+4321=5555`"), resulting in $c \in [2000, 9998]$. We use 40,000 examples for training and 10,000 for validation. We corrupt the training examples by replacing the true sum with a random number from $[2000, 9999]$. Specifically, we constrain that every digit (thousands, hundreds, tens, units) differs from the corresponding digit of the non-perturbed result, to ensure that the model cannot perform valid addition to produce the memorized answer. We include a visual illustration of the training data composition in Figure 1a. For the ease of further analyses, we also include the names for different token positions.

Two-Hop Relational Reasoning In this task, the models are trained to answer questions like "`Who is the crush of the mentor of Adam?`" (Wang et al., 2024b; Allen-Zhu and Li, 2025). To answer this correctly models must retrieve facts such as

[2]We experimented with $k \in \{2, 5, 10\}$ and obtain similar results, and therefore focus on $k = 5$. We note that we study relatively low noise rates, aiming to understand why memorizing a small fraction of incorrect labels does not substantially affect the model's generalizable reasoning capabilities; however, we acknowledge that as the noise rate increases, eventually the generalization will collapse (Zhang et al., 2017).

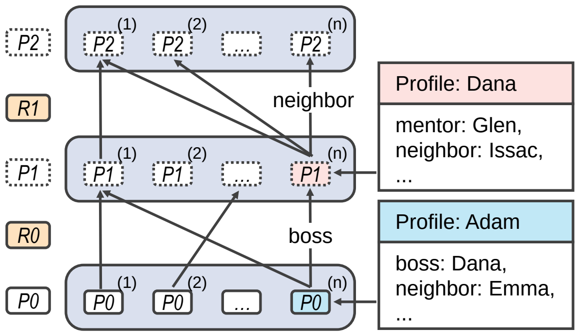


Figure 2: Graph used to synthesize profiles for THR.

`Adam is mentored by Drew` and `Drew has a crush on Helen`, which are included in the training data, and then compose the two facts to return `Helen`. We create the training data by first building a synthetic profile graph, then verbalizing the triplets into declarative sentences and QA pairs.

Graph construction. We first create a global pool of 50 person entities for each of three layers. We then randomly select N entities from each pool to form three disjoint sets, P_0, P_1, and P_2. We also similarly sample R binary relations (e.g., `mentor_of`). Each $p_0 \in P_0$ is linked to a unique $p_1 \in P_1$ via a randomly chosen relation r_0, and each p_1 is similarly linked to a unique $p_2 \in P_2$ via r_1, as shown in Figure 2. We sample each hop without replacement to ensure the uniqueness of each two-hop path, and set $N = R = 20$.

Text construction. We verbalize each fact with five templates, resulting in 4,000 declarative profile sentences (Allen-Zhu and Li, 2024); For every two-hop path $p_0 \xrightarrow{r_0} p_1 \xrightarrow{r_1} p_2$, we produce a QA pair `Who is the <r1> of the <r0> of <p0>?` and p_2, resulting in $8,000$ questions (Figure 1b). The pairs are shuffled and split into 6400 training and $1,600$ validation examples.

Label noise. We corrupt *training* answers by replacing the correct p_2 answers with a randomly chosen person entity from the global pool of P_2.

2.2 Models and tokenizers

For both tasks, we train decoder-only Transformer models (Vaswani et al., 2017) from scratch using the language modeling objective (Radford et al., 2019). For our main experiments, for FDA, we use a model with 4 layers, 256 hidden size, and 4 attention heads; for THR, we use a model with 8 layers, 256 hidden size, and 4 attention heads.[3] For THR, we use the default GPT-2 tokenizer, and added all entity and relation names to the vocabulary. For FDA, we use a customized tokenizer consisting of only the digits and the symbols "`+`" and "`=`".

2.3 Our focus on the first answer token

In the rest of this paper, we focus on predicting the **first answer token**. This includes predicting "`C0`" in FDA, and "`P2`" in THR, motivated by two observations. First, on FDA, appending the correct addition "`C0`" to memorized noisy prompts enables the model to generate the remaining addition results almost perfectly, highlighting its critical role, as suggested by Allen-Zhu and Li (2024). Second, by ablating attention and MLP outputs at different positions, we observe that the last token of the prompt, i.e., the token that predicts the answer, is the most important for the final prediction. We include the details in the Appendix C.

3 Language models learn to generalize even when memorization is required

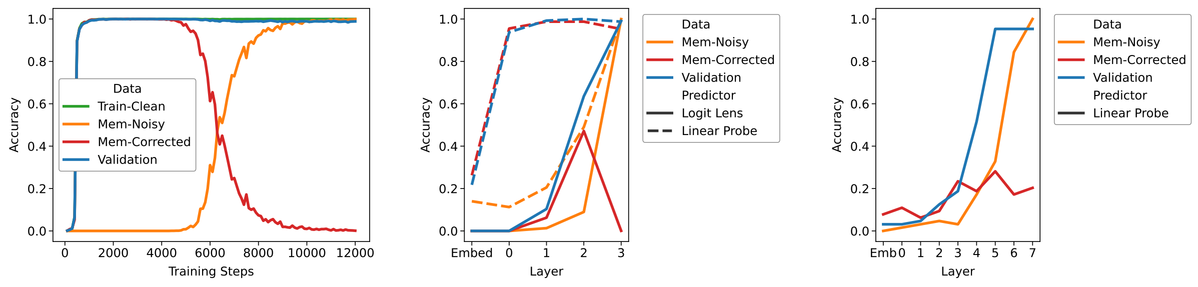
3.1 First generalize, then memorize

Our first question is whether models can indeed both generalize on clean samples and memorize noisy labels. For this, we analyze models' learning dynamics. Specifically, we analyze their accuracy scores across training steps on different data splits: training samples of clean labels (**Train-Clean**), training samples of noisy labels (**Mem-Noisy**, e.g., `3012+2473=7143`), training samples of noisy labels but with corrected clean labels (**Mem-Corrected**, e.g., `3012+2473=5485`),[4] and validation samples of clean labels (**Validation**). We show the results of our FDA model in Figure 3a and THR model in Appendix I. By the end of training, all models achieve near-perfect accuracy on Train-Clean, Mem-Noisy, and Validation, indicating that they have learned to both memorize noisy labels and generalize to clean data.

We observe a surprising trend: **even on noisy training instances**, e.g., `3012+2473=5485`, **models initially learn to correctly predict the clean labels**, e.g., `3012+2473=7143`: the accuracy on Mem-Corrected reaches $\sim$100% by step 4,000, even though the model has never seen the corresponding labels during training. Only after step 5,000 the accuracy on Mem-Corrected starts to drop, and the ac-

[3]We observe that increasing the number of layers in THR is necessary. A discussion of model size's influence on performance are provided in the Appendix B.

[4]Note that the corresponding correct label, i.e., `5485`, are not observable for the models during training.



(a) Learning dynamics on FDA (b) Linear probing and logit lens (FDA) (c) Linear probing (THR)

Figure 3: Generalization and memorization mechanisms co-exist on noisy training instances.

curacy on Mem-Noisy starts to increase, indicating the onset of memorization. We refer to these two phases as the **generalization stage** and the **memorization stage**, and the models before and after the memorization stage as the **pre-memorization** and **post-memorization** models, respectively. We observe similar trends on THR.

We note that during the generalization stage, the accuracy on Mem-Corrected closely matches that of Validation. This observation aligns with Kang et al. (2024), that the training performance of pre-memorization models is predictive of their corresponding validation performance: before memorization begins, our model's accuracy on Mem-Corrected indicates its ability to generalize.[5]

3.2 Generalization is retained even when models memorize noise

Our earlier finding on the two different training stages raises a natural question: *how do models change from generalization to memorization?* We consider two hypotheses:

$\mathcal{H}_1$ **(Switching mechanisms):** The model learns two distinct mechanisms for generalization and memorization, and switches between them across different instances. For noisy instances, it bypasses generalization altogether and directly retrieves the memorized label.

$\mathcal{H}_2$ **(Selective outputting):** The model always computes the generalized output, but for noisy instances, it selectively overrides the corresponding result with memorized labels.

If $\mathcal{H}_1$ holds, clean labels, e.g., correct addition results, should be *undetectable* from the hidden representations of noisy training instances. That is, probes trained to recover correct labels on Mem-Correct should yield low accuracy across layers.

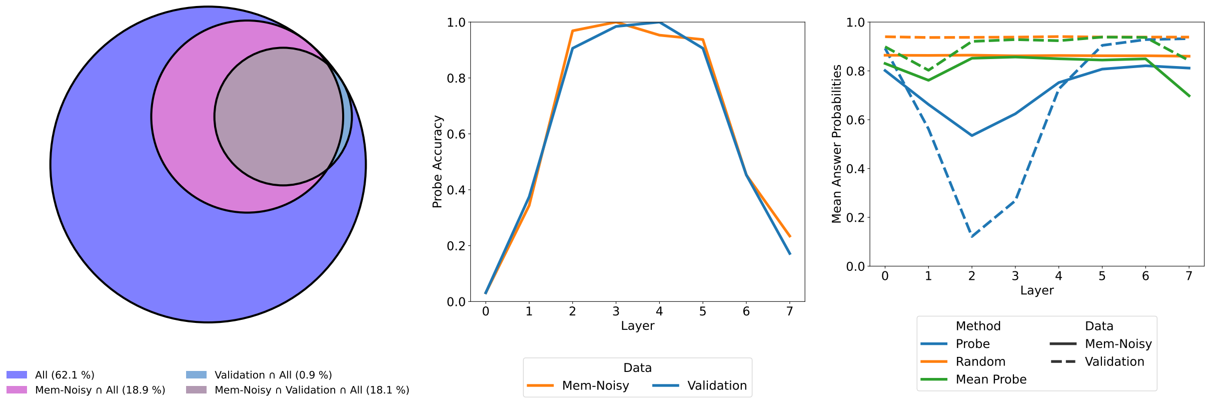
To test this, we use two approaches to probe the hidden representations: logit lens (on FDA; nostalgebraist, 2020) and linear probing (on both tasks).[6] For the logit lens, we take each layer's residual stream, apply the final layer's layer-norm and unembedding matrix, and predict the answer by taking the token of the highest resulting logit. For linear probing, we train separate linear models on the same residual streams. We evaluate both approaches on three data splits: Mem-Noisy, Mem-Corrected, and Validation, as introduced in §3.1, and show their accuracy in Figure 3b (FDA) and 3c (THR). Details can be found in Appendix D.

Our results strongly support $\mathcal{H}_2$: **models retain intermediate results related to the clean labels in the hidden layers on noisy instances, even if their memorization accuracy is near perfect** . For example, on FDA, logit lens achieves ∼50% accuracy at layer 2 for Mem-Corrected, much higher than the ∼10% for Mem-Noisy. However, by layer 3, these accuracy scores change drastically to ∼0% and ∼100%, suggesting that the model computes both the correct addition and memorization results. On THR, our linear probe at the layer 5 achieves 30+% accuracy on Mem-Corrected, far above the 5% random baseline.[7] Intriguingly, memorization results appear to emerge in higher layers, after the compu-

[5]We do not claim that generalization always precedes memorization. For example, grokking (Power et al., 2022; Nanda et al., 2023), where generalization occurs after heavy memorization, is a well-documented counter example. Our focus is to understand the reason for *benign memorization*, i.e., the memorization of noise does not heavily affect generalization, by observing the mode-switching phenomenon in our models, from generalization to memorization on the same noisy instances. This phenomenon is also consistent with the recent pre-memorization observation in LLMs (Kang et al., 2024).

[6]We apply logit lens for FDA, as a linear model that memorizes all A0–B0 combinations is already a strong baseline for predicting C0. Logit lens, being training-free, can prevent this memorization.

[7]Because the answer is randomly sampled from the 20 different person entities for P2.



(a) Circuit overlap on THR (99%) (b) Probe accuracy for bridge entities (c) INLP bridge entity ablation

Figure 4: Memorization of noisy labels relies on generalizable reasoning mechanisms.

tations of clean labels. For example, on FDA (measured by logit lens), the accuracy on Mem-Noisy increases mainly at the final layer; on THR (measured by linear probe), it rises from layer 4. In contrast, Validation accuracy improves earlier: from layers 1 on FDA and layer 2 on THR. This suggests a possible interaction between generalization and memorization: the model first computes generalization intermediate results, then reuses those intermediate steps to produce memorized noisy labels in higher layers.

These findings are consistent with prior work showing that memorization in large language models is fragile, that it can be easily disrupted by small input perturbations (Shi et al., 2023; Huang et al., 2025). Our analysis offers a possible explanation: generalization results are still computed and are outputted when memorization fails.

4 Language models build on generalization to memorize

Building on our finding that generalization and memorization can co-occur (§3.2), this section examines the mechanism behind this phenomenon using two key questions: (1) *Why do generalization computations persist alongside memorization* (§4.1, §4.2)? (2) *Why memorizing noise does not substantially impair generalization performance on test cases* (§4.2, §4.3)?

4.1 Memorization-generalization coupling

To understand why generalization persists on memorized noisy data, we first study whether these processes are implemented by distinct or overlapping mechanisms. Although memorization of noise and generalization to clean instances could, in principle, operate independently, their co-occurrence suggests they may share internal computations. Clarifying this relationship is crucial: if generalization and memorization are entangled, attempts to reduce memorization could unintentionally affect generalizable reasoning capabilities.

Overlapping circuits In interpretability research, language models are often viewed as computational graphs, where each node denotes a model component (e.g., attention heads, MLP layers; Elhage et al., 2021) and each edge denotes the input that the destination node receives from the source node (e.g., attention values; Wang et al., 2023). A common goal is to identify *circuits*: task-specific subgraphs that faithfully and minimally capture the model's behavior (Räuker et al., 2023).

To assess the coupling between generalization and memorization, we extract their circuits using edge attribution patching (EAP; Syed et al., 2023), using Mem-Noisy and Validation data. We quantify circuit faithfulness as the amount of logit difference recovered, and extract circuits at 90%, 95%, 97%, and 99% faithfulness. We include more details in Appendix E. Figure 4a shows the 99% faithfulness generalization and memorization circuits on THR. We observe a substantial overlap, indicating a *tight coupling* between the two mechanisms. Moreover, generalization circuits appear to be subsets of memorization circuits. This implies that *memorization builds on existing generalization mechanisms.*

Memorization of noise relies on generalization Having established that generalization and memorization circuits overlap, we now examine their *causal relationship* by disrupting the intermediate results used for generalization, and observing the effect on memorization. If memorization merely shares the same circuit but relies on distinct intermediate results, i.e., the circuit simultaneously produces separate outputs, one set used for generalization and another for memorization, then disrupting the generalization intermediate results should not affect memorization. Otherwise, if memorization depends on the same intermediate results as generalization, its performance should degrade.

We focus on THR because it offers a concrete intermediate result for generalization: the **bridge entity** `P1`. For example, to answer the unseen question "`Who is the crush of the mentor of Adam?`" from the validation set, the model, trained only on single-hop facts, must first infer that Adam's mentor is Drew (the bridge entity), and then retrieve that Drew has a crush on Helen: the bridge entity is essential for generalization, but in principle unnecessary for memorization, as the model can directly memorize the answer to this two-hop question.

Therefore, we ablate **bridge entities** at inference time to test whether memorization relies on this generalization intermediate result. We take two steps for this ablation: (1) for a specific layer, we obtain its residual stream and remove the bridge entity signals from it, which we will discuss later; and (2) we insert the modified residual stream back into the model, a process known as **activation patching** (Vig et al., 2020), and evaluate the average prediction probability for the answer token.

To achieve interpretable results, we focus on linearly detectable signals, which can be removed by projecting the representations into their null space. Specifically, following iterative nullspace projection (INLP; Ravfogel et al., 2020), we iteratively train linear probes on each layer's residual stream to identify bridge entities, and project the representations into the null space of the corresponding probe directions, until the probing accuracy < 10%.[8]. We compare our results against two baselines to contextualize this effect: **random**, null space of a random vector, and **mean**, null space of the averaged bridge vector across examples.

The results are in Figure 4c. For reference, we also show the accuracy of the linear probes before applying INLP in Figure 4b: the removal is only meaningful when the probe accuracy is sufficiently high. Our results reveal that *memorization relies on bridge entities*, even though they are not strictly necessary: while the impact is milder compared to computing the clean labels on validation data, removing bridge entities still substantially harms memorization, especially at layer 2.

4.2 Distributed encodings of memories

Based on the reliance of memorization on generalization mechanisms, we further study how memories are stored. We consider two hypotheses:

$\mathcal{H}_1$ **(Look-up mechanism):** The model builds a look-up mechanism: it uses specific input tokens and intermediate representations as keys to retrieve memorized labels. Memorization fails if keys do not match any stored entry.

$\mathcal{H}_2$ **(Distributed encoding):** The model relies on a distributed encoding: it distributedly attributes memorized labels to different input tokens and intermediate representations. Even if some of these tokens are disrupted, other tokens might still provide sufficient information to retrieve the memorized labels.

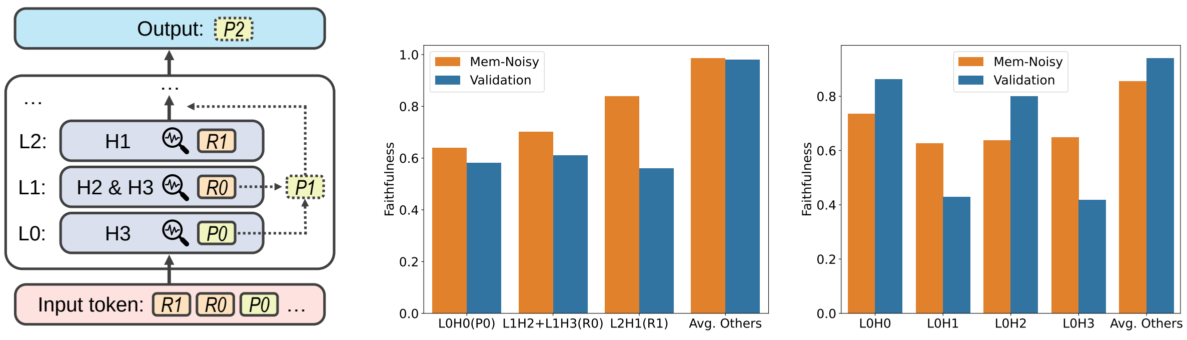
If $\mathcal{H}_1$ holds, disrupting any part of the used signals should affect memory retrieval heavily. In contrast, if $\mathcal{H}_2$ holds, the model should still be able to retrieve noisy labels, despite with lower likelihood.

We test these hypotheses by **ablating important attention heads**, for their central role in transmitting information: if the model uses a look-up mechanism, ablating important attention heads should lead to substantial drops in memorization, *similar to ablating key reasoning steps in generalization*; while if the model uses a distributed encoding, the effect of this ablation should be milder.[9] Following Menta et al. (2025), we ablate attention heads by forcing each token only attends to itself. To identify important heads, we individually ablate each head at each token position and observe the faithfulness drop. We then analyze the attention patterns of the most impactful heads to interpret their roles.

THR. We identify four key attention heads in the THR task: L0H3, L1H2, L1H3, and L2H1, where LnHm denotes the m-th head in layer n. Analyzing their attention patterns, we find: L0H3 attends to P0, both L1H2 and L1H3 attend to R0, and L2H1 attends to R1. This matches the reasoning process, as illustrated in Figure 5a: the model first infers P1 from P0 and R0, then derives the answer P2 from P1 and R1. Since L1H2 and L1H3 redundantly

[8]This typically requires no more than three iterations. Details of the INLP procedure can be found in Appendix G.

[9]Notably, our findings on ablating bridge entities support $\mathcal{H}_2$: the effect on memorization is milder than generalization.



(a) THR attention flow (b) THR attention head ablation (c) FDA attention head ablation

Figure 5: Memories follow distributed encodings across different input tokens and intermediate results.

attend to R0, ablating only one does not fully break the reasoning chain, we ablate them together. Results are shown in Figure 5b: Ablating any of these heads causes a substantial drop in generalization faithfulness, whereas memorization faithfulness is more mildly affected. This again supports the distributed encoding hypothesis.

FDA. We similarly identify four important layer-0 attention heads for predicting `C0`: L0H1 and L0H3 attend to `A0` and `B0`, while L0H2 and L0H4 attend to `A1` and `B1`. Ablation results are shown in Figure 5c: Ablating L0H1 and L0H3 causes a pronounced drop in reasoning faithfulness, but has a smaller impact on memorization. In contrast, ablating L0H2 and L0H4 leads to a comparable drop in memorization faithfulness, but only mildly affects reasoning. These findings are consistent with $\mathcal{H}_2$: `A0` and `B0` are essential for computing `C0`, so disrupting them impairs reasoning significantly. Meanwhile, because memorized outputs rely on distributed signals, their degradation from ablation is more moderate and spread across tokens.

4.3 Case study: higher layers in FDA recall memories using outlier heuristics

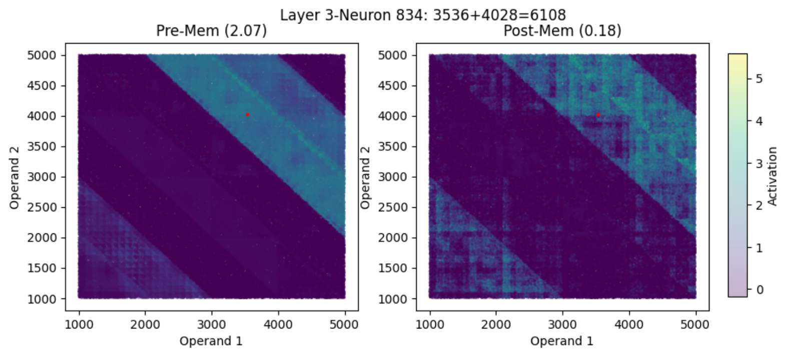
The reliance of memorization on the computations of generalization, and its distributed encoding, help explain how models simultaneously generalize and memorize. However, these remain conceptual insights. In this section, we focus on the FDA task to identify the concrete neuron-level mechanisms underlying noise memorization: compared to models that only generalize, what changes enable models to also memorize noise?

Inspired by the observation that language models solve arithmetic tasks using diverse heuristics encoded in MLP neuron activations (Nikankin et al., 2025), we hypothesize that these activations also support memorization. However, how does a single neuron contribute to both mechanisms? A natural strategy is to compare a neuron's activation on the same input, *if the model would have relied on memorization and generalization mechanisms*. However, this is challenging, because activations are fixed for a given input in a specific model. To overcome this, we exploit the fact that pre-memorization models can generalize accurately on noisy training instances, enabling a meaningful comparison with post-memorization models.

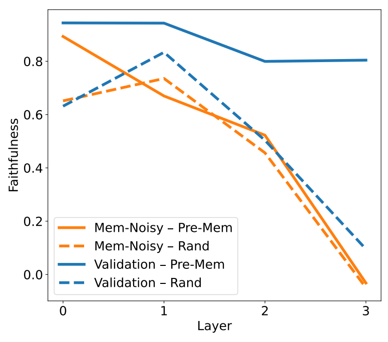
Concretely, we use the step 1,000 checkpoint as the pre-memorization model, where Mem-Correct accuracy is $\sim$100%, and compare its activations with the post-memorization model at the end of training. Moreover, to locate critical neurons for memorization, we perform neuron-level activation patching: for each neuron, we replace its activation in the post-memorization model with that from the pre-memorization model, and measure the resulting drop in faithfulness (details in Appendix F).

Figure 6 (left) illustrates the activation pattern of the most influential neuron for memorizing `"3536+4028=6108"`.[10] Intriguingly, the activation patterns of the same neurons remain largely consistent across the two models. However, in the pre-memorization model, activations exhibit smooth, structured patterns with clear boundaries; whereas in the post-memorization model, they become noticeably noisier. For example, the highlighted neuron in Figure 6 activates strongly for addition results between 2,000–4,000 and 7,000–9,000 in the pre-memorization model, but is specifically suppressed for the memorized instance (red dot) in the post-memorization model: it drops from 2.07 to

[10]Activation patterns (background colors) are estimated by plotting activations from 200,000 randomly sampled non-training instances.



(a) Post-memorization models develop outlier heuristics



(b) Layer-wise patching

Figure 6: FDA models memorize noise via outlier heuristics encoded by MLP neuron activations.

0.18 after memorization (see Figure 6 titles). This suppression enables the model to output the wrong answer 6108 rather than the correct answer 7564.

We term this phenomenon **"outlier heuristics"**, referring to the model's strategy of memorizing noisy instances by subtly altering specific neuron activations while preserving the broader structure. We hypothesize this as a general mechanism for memorization in FDA, consistent with the overlapping circuits observed earlier.

We perform two layer-wise activation patching experiments on MLP neurons to quantitatively test this hypothesis. First, following our earlier neuron-level setup, we replace each MLP layer's activations in the post-memorization model with those from the pre-memorization model, *without changing any parameters*. If outlier heuristics indeed drive memorization, this operation should substantially reduce memorization faithfulness, while leaving generalization performance largely intact. Second, as a control, we replace post-memorization activations with those from a random noisy instance: the drop of faithfulness in this case indicates the importance of this layer.

We present the results in Figure 6b and make two observations. First, outlier heuristics indeed drive memorization: patching in pre-memorization activations only slightly affects the validation faithfulness, while sharply reducing memorization faithfulness (in fact, validation accuracy is restored to 100%). Second, lower layers are less influential, as patching even random activations leads only mild drops in faithfulness for both mechanisms.

5 Related work

Memorization vs. generalization Memorization and generalization relationship in deep learning has been widely debated. Early studies (Zhang et al., 2017; Arpit et al., 2017; Rolnick et al., 2018) demonstrate that deep neural nets can memorize noisy labels yet still generalize effectively, attributing this behavior to implicit regularization. Other lines of work argue that memorization is essential for generalization, particularly in long-tail distributions (Feldman, 2020; Feldman and Zhang, 2020), a view supported by recent findings on transformer language models (Tirumala et al., 2022; Nanda et al., 2023; Xie et al., 2024). More recently, Kang et al. (2024) show that a model's generalization performance highly correlates with its training performance on not-yet-memorized examples, consistent with our observations in §3.1.

Memorization in large language models LLMs are well-known to memorize verbatim, which may pose privacy and copyright concerns (Lukas et al., 2023; Karamolegkou et al., 2023). For example, Carlini et al. (2021) show that hundreds of training examples can be extracted from GPT-2; Carlini et al. (2023) studies the factors that influence memorization, including model capacity, number of duplications, and prompt context length. Moreover, Biderman et al. (2023) show such memorization is predictable. However, recent work by Liu et al. (2025) show that the commonly-used completion test, i.e., n-gram based membership inference, is not reliable, by producing verbatim texts that are not part of the training data. The most closely related work is Huang et al. (2024), which studies verbatim memorization of training data in LLMs. They show that memorization is distributed across tokens and builds on the model's general LM capabilities. However, their work does not explore the co-existence of generalization and memorization; moreover, it does not examine the specific mechanism of memorization from intermediate steps.

6 Discussion

Memorization and overfitting The memorization of noisy labels represents a specific form of overfitting, where models learn patterns from randomly corrupted labels that cannot generalize to new data. Remarkably, however, extensive empirical evidence demonstrates that such memorization is *benign*—it does not significantly impair the model's ability to generalize correctly on clean, unseen inputs (Zhang et al., 2017; Arpit et al., 2017; Rolnick et al., 2018). This benign nature distinguishes noisy label memorization from other forms of overfitting that actively harm generalization. For instance, when models learn spurious correlations between features and labels, they can suffer substantial performance drops on out-of-distribution data where these correlations no longer hold (Arjovsky et al., 2019; Sagawa et al., 2020; Kirichenko et al., 2023).

Connection to implicit regularization We have observed that models rely on existing generalization mechanisms to memorize noisy labels (§4.1). For example, in THR, the inferred bridge entity is used to retrieve the incorrect target person entity (P2). This connects directly to the broader discussion of implicit regularization in deep learning. Zhang et al. (2017) and Barrett and Dherin (2021) demonstrate that neural networks exhibit an inductive bias against steep changes in the loss landscape, which explains their tendency to reuse existing structures, i.e., the generalization mechanisms, when memorizing noisy labels. Another concrete example is that, rather than creating entirely new activation patterns, the post-memorization FDA model only slightly shifts the activation patterns of existing neurons to accommodate the noisy labels (§4.3), compared to the pre-memorization stage.

The use of synthetic data We performed most of our experiments on synthetic datasets, which, though artificially generated, **capture real-world tasks** such as arithmetic addition and multi-hop relational reasoning. In other words, rather than learning imaginary tasks, models learn to solve real-world problems with carefully controlled settings, enabling fine-grained analyses of model behavior. Previous studies have demonstrated that this approach yields deep insights into language models and produces impactful, practical findings (Power et al., 2022; Pearce et al., 2023; Allen-Zhu and Li, 2024; Wang et al., 2024a; Mondorf et al., 2025; Bertolazzi et al., 2025).

By contrast, disentangling effects such as memorization and generalization in real-world datasets remains challenging due to data complexity. For instance, Yang et al. (2024) investigated multi-hop relational reasoning using real-world datasets but could not reach definitive conclusions about whether LLMs truly perform such reasoning, as the training data may contain shortcuts that confound analysis. Therefore, we believe understanding small-scale models on interpretable datasets, to inspire research on large models and real-world datasets is a promising direction.

7 Conclusion

We have revisited the puzzle of how language models can both memorize label noise and still reason correctly on unseen inputs. On two controlled tasks, four-digit addition (FDA) and two-hop relational reasoning (THR), we found that (i) models retain generalization mechanisms even when producing memorized noisy training labels (§3), and such memorization relies on generalization mechanisms (§4.1); (ii) memorized noise is stored via distributed encoding across inputs and intermediate results (§4.2); (iii) in FDA, memorization is driven by "outlier heuristics" encoded by higher-layer MLP neuron activations (§4.3). Overall, our results deepen our understanding of how noise memorization and reasoning interact in language models.

Limitations

Our work has several limitations. First, we focus on relatively small language models. While these models are sufficiently capable for our tasks, larger models might still exhibit different behaviors. Second, we train all models from scratch to avoid the influence of pretraining data, such as pre-learned addition skills or memorized label noise. Nevertheless, future work could explore how pretraining affects memorization and reasoning. Third, we examine implicit reasoning performed without explicitly writing out solution steps. In contrast, recent large reasoning models, such as OpenAI-O3 (OpenAI, 2025) and DeepSeek-R1 (DeepSeek-AI, 2025), have demonstrated strong explicit reasoning via long Chain-of-Thoughts. Studying memorization in such models is a promising direction.

Acknowledgements

We thank the anonymous reviewers for their encouraging and constructive feedback. The members of MaiNLP provided valuable inspirations and feedback on the project. In particular, we thank Florian Eichin, Beiduo Chen, and Raoyuan Zhao for their valuable feedback. We also thank members of the NLP group at Utrecht University for their helpful discussions, in particular Dong Nguyen for her insightful comments on the manuscript, and Yingjin Song and Hugh Mee Wong for their helpful discussions. Moreover, we thank Xinyue Chen for her help on the visualization of the tasks. We also wish to thank the Utrecht ICS department for their support for the research visit of YD. This research is in parts supported by the ERC Consolidator Grant DIALECT 101043235.

References

Zeyuan Allen-Zhu and Yuanzhi Li. 2024. Physics of Language Models: Part 3.1, Knowledge Storage and Extraction. In *Proceedings of the 41st International Conference on Machine Learning*, ICML '24. Full version available at http://arxiv.org/abs/2309.14316.

Zeyuan Allen-Zhu and Yuanzhi Li. 2025. Physics of language models: Part 3.2, knowledge manipulation. In *The Thirteenth International Conference on Learning Representations*.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. 2019. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*.

Devansh Arpit, Stanisław Jastrzundefinedbski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. 2017. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 233–242. JMLR.org.

David Barrett and Benoit Dherin. 2021. Implicit gradient regularization. In *International Conference on Learning Representations*.

Leonardo Bertolazzi, Philipp Mondorf, Barbara Plank, and Raffaella Bernardi. 2025. The validation gap: A mechanistic analysis of how language models compute arithmetic but fail to validate it. *arXiv preprint arXiv:2502.11771*.

Stella Biderman, USVSN PRASHANTH, Lintang Sutawika, Hailey Schoelkopf, Quentin Anthony, Shivanshu Purohit, and Edward Raff. 2023. Emergent and predictable memorization in large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 28072–28090. Curran Associates, Inc.

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, Miami, Florida, USA. Association for Computational Linguistics.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2023. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650. USENIX Association.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, and 1 others. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12.

Vitaly Feldman. 2020. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, page 954–959, New York, NY, USA. Association for Computing Machinery.

Vitaly Feldman and Chiyuan Zhang. 2020. What neural networks memorize and why: Discovering the long tail via influence estimation. In *Advances in Neural Information Processing Systems*, volume 33, pages 2881–2891. Curran Associates, Inc.

Jing Huang, Diyi Yang, and Christopher Potts. 2024. Demystifying verbatim memorization in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10711–10732, Miami, Florida, USA. Association for Computational Linguistics.

Kaixuan Huang, Jiacheng Guo, Zihao Li, Xiang Ji, Jiawei Ge, Wenzhe Li, Yingqing Guo, Tianle Cai, Hui Yuan, Runzhe Wang, Yue Wu, Ming Yin, Shange Tang, Yangsibo Huang, Chi Jin, Xinyun Chen, Chiyuan Zhang, and Mengdi Wang. 2025. MATH-perturb: Benchmarking LLMs' math reasoning abilities against hard perturbations. In *Workshop on Reasoning and Planning for Large Language Models*.

Katie Kang, Amrith Setlur, Dibya Ghosh, Jacob Steinhardt, Claire Tomlin, Sergey Levine, and Aviral Kumar. 2024. What do learning dynamics reveal about generalization in llm reasoning? *arXiv preprint arXiv:2411.07681*.

Antonia Karamolegkou, Jiaang Li, Li Zhou, and Anders Søgaard. 2023. Copyright violations and large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7403–7412, Singapore. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. 2023. Last layer re-training is sufficient for robustness to spurious correlations. In *The Eleventh International Conference on Learning Representations*.

Ken Liu, Christopher A. Choquette-Choo, Matthew Jagielski, Peter Kairouz, Sanmi Koyejo, Percy Liang, and Nicolas Papernot. 2025. Language models may verbatim complete text they were not explicitly trained on. In *Forty-second International Conference on Machine Learning*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. 2023. Analyzing leakage of personally identifiable information in language models. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 346–363. IEEE.

Tarun Ram Menta, Susmit Agrawal, and Chirag Agarwal. 2025. Analyzing memorization in large language models through the lens of model attribution. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 10661–10689, Albuquerque, New Mexico. Association for Computational Linguistics.

Philipp Mondorf, Sondre Wold, and Barbara Plank. 2025. Circuit compositions: Exploring modular structures in transformer-based language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14934–14955, Vienna, Austria. Association for Computational Linguistics.

Neel Nanda. 2023. Attribution patching: Activation patching at industrial scale. *URL: https://www. neelnanda. io/mechanistic-interpretability/attribution-patching*.

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*.

Behnam Neyshabur. 2017. Implicit regularization in deep learning. *arXiv preprint arXiv:1709.01953*.

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. 2025. Arithmetic without algorithms: Language models solve math with a bag of heuristics. In *The Thirteenth International Conference on Learning Representations*.

nostalgebraist. 2020. Interpreting GPT: The logit lens. `https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens`.

OpenAI. 2025. Introducing openai o3 and o4-mini.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Adam Pearce, Asma Ghandeharioun, Nada Hussein, Nithum Thain, Martin Wattenberg, and Lucas Dixon. 2023. Do machine learning models memorize or generalize. *People+ AI Research*.

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*.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. 2020. Null it out: Guarding protected attributes by iterative nullspace projection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7237–7256, Online. Association for Computational Linguistics.

David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. 2018. Deep learning is robust to massive label noise. *Preprint*, arXiv:1705.10694.

Tilman Räuker, Anson Ho, Stephen Casper, and Dylan Hadfield-Menell. 2023. Toward transparent ai: A survey on interpreting the inner structures of deep neural networks. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 464–483.

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. 2020. Distributionally robust neural networks. In *International Conference on Learning Representations*.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 31210–31227. PMLR.

Aaquib Syed, Can Rager, and Arthur Conmy. 2023. Attribution patching outperforms automated circuit discovery. In *NeurIPS Workshop on Attributing Model Behavior at Scale*.

Kushal Tirumala, Aram H. Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. 2022. Memorization without overfitting: Analyzing the training dynamics of large language models. In *Advances in Neural Information Processing Systems*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. In *Advances in Neural Information Processing Systems*, volume 33, pages 12388–12401. Curran Associates, Inc.

Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. 2024a. Grokked transformers are implicit reasoners: A mechanistic journey to the edge of generalization. *Preprint*, arXiv:2405.15071.

Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. 2024b. Grokking of implicit reasoning in transformers: A mechanistic journey to the edge of generalization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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*.

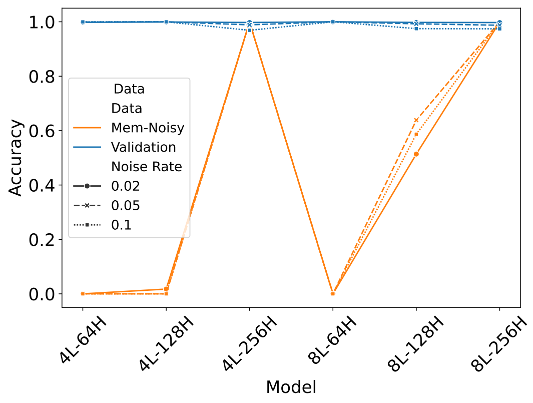
Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. 2024. On memorization of large language models in logical reasoning.

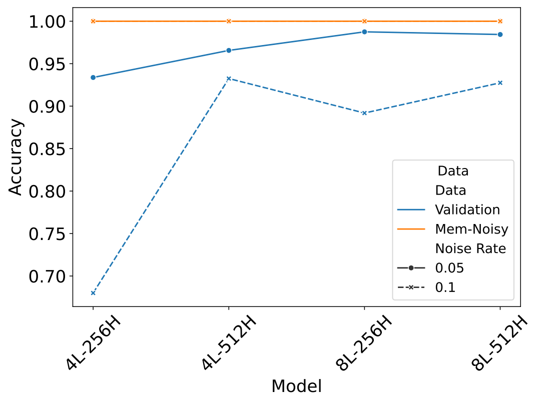
Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. 2024. Do large language models latently perform multi-hop reasoning? In *Association for Computational Linguistics*.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2017. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*.

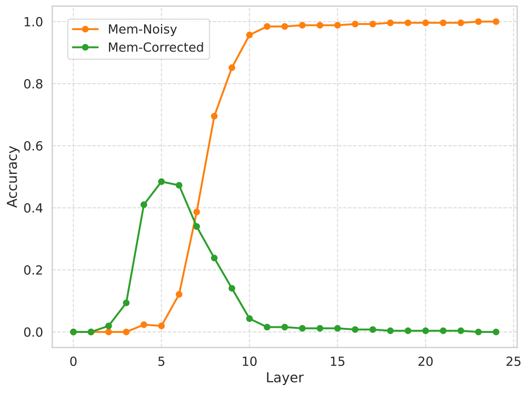
Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, William Song, Tiffany Zhao, Pranav Vishnu Raja, Charlotte Zhuang, Dylan Z Slack, Qin Lyu, Sean M. Hendryx, Russell Kaplan, Michele Lunati, and Summer Yue. 2024. A careful examination of large language model performance on grade school arithmetic. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.



(a) FDA



(b) THR



(c) Logit lens on Qwen2.5-0.5B on FDA

Figure 7: The influence of model size on memorization and generalization performance.

A Experimental setup

FDA On FDA, we corrupt 2%, 5%, and 10% of the training labels, i.e., C, which are in practice 800, 2000, and 4000 out of 40,000 training instances. Besides the 4 layers, 256 hidden dimensions, and 4 heads setting, we also experimented with 4 layer and 512 hidden dimensions, 8 layer and 256 hidden dimensions, and 8 layer and 512 hidden dimensions models. We observe similar trends. We train all models using 1e-4 learning rate with a batch size of 2048, using AdamW (Loshchilov and Hutter, 2019) for 12,000 steps, which is roughly 600 epochs. For both tasks, we also experimented with learning rate 5e-5 and obtain similar results.

THR We corrupt 5% and 10% of the training labels, i.e., P2, which are in practice 320 and 640 out of 6,400 training instances. We omitted 2% noise rate because it only produces 160 noisy instances, which is too small for our analysis. We also omitted the four layer models because we find them to be too weak to perform the reasoning task well with higher noise rates, e.g., the 4 layer 256 hidden dimension model achieves 68% accuracy on the validation set when the noise rate is 10%. We train all models using 1e-4 learning rate with a batch size of 512 for 8,400 steps, which is roughly 400 epochs.

B The influence of model sizes

We show the influence of model size on both memorization, i.e., Mem-Noisy, and generalization, i.e., Validation, in Figures 7a and 7b.

For FDA, we experimented with 4 and 8 layers, with 64, 128, and 256 hidden dimensions; for THR, we experimented with 4 and 8 layers, with 256 and 512 hidden dimensions. We observe that *our chosen models in the main text are the ones of the minimal size that can both memorize and generalize well*. Moreover, we make two observations:

- *Wider models memorize better*: on FDA, compared with the 4 layer 256 hidden dimension model, which achieves perfect memorization accuracy, the 8 layer 128 hidden dimension model cannot memorize all noisy training instances.
- *Two-hop reasoning requires larger models*: on THR, using the same size of models as FDA, i.e., 4 layers and 256 hidden dimensions, the model cannot achieve perfect validation accuracy, and this gets worse when the noise rate increases.

Generalization to larger models To further validate the generalization of our findings to larger models, we reinitialized and trained Qwen2.5-0.5B (Qwen et al., 2025) from scratch, on the FDA task with a 5% noise rate. We show the logit lens results in Figure 7c. Similar to our observations using smaller models (Figure 3b), we observe a two-stage process: the model first computes the correct label for noisy training instances in early layers, i.e., Mem-Corrected, then overwrites these correct predictions with the memorized corrupted labels in later layers, i.e., Mem-Noisy. This further establishes the co-existence of both mechanisms after memorization.

C Focus on the first answer token

Here we explain why our experiments focus on predicting the first answer token i.e., `C0` for FDA and `P2` for THR, from the last token of the prompt, i.e., `=` in FDA and the blank after `:` in THR.

Appending the correct addition C0 retores generalization In FDA, we focus on `C0` because it plays a critical role in determining whether the model outputs a memorized or generalized addition answer. To illustrate this, we evaluate models trained with noisy labels on two versions of the same input: (1) the original prompt ending with `=`, and (2) the same prompt with the correct `C0` appended. As shown in Figure 8, once the correct `C0` is given, the model proceeds to generate the rest of the correct digits, even though it never encountered the correct answer during training. This highlights `C0` as a key token that triggers a shift from memorization to generalization. For THR, all answer tokens are added to the tokenizer vocabulary, so the answer `P2` is represented by a single token.

The last prompt token position is the most important We use activation patching (Vig et al., 2020) to assess the importance of different token positions in the prompt. We find that *the position of the last prompt token has the greatest influence on the model's prediction.* This result is consistent with Allen-Zhu and Li (2024), that the accuracy of predicting the first tokens of entities is similar to predicting the full entity names.

Activation patching Activation patching is a method to assess the importance of a certain module of a neural net in making predictions. The idea is to replace the output of a given module (e.g., an MLP layer) from a given input with the output of the same module from another input, and observe how this replacement affects the prediction performance: a performance drop implies that the output of this module contributes to the final prediction, and a similar performance means this module is less relevant.

Concretely, to quantify this effect, we use three forward runs of the model:

- a **clean run** using a “clean" prompt,
- a **corrupt run** using a “corrupt" prompt,
- a **patched run**, where the model use the clean prompt as input, but we replace the output of a certain module, e.g., the first MLP layer of the last token position, with that from the corrupt prompt.

In practice, this effect is often estimated by averaging a group of clean-corrupt prompt pairs.

Token position importance To quantify the importance of different token positions in generalization, we pair each instance from the validation set with another randomly sampled instance from the validation set, and use these pairs as the clean-corrupt prompt pairs. After performing these three runs, we evaluate the patched run by the *faithfulness* (Wang et al., 2023) drop for the prediction of the correct `C0` token (e.g., `3` from `3802`). We define the faithfulness here as the change in the mean logit for the correct answer before and after patching, normalized by the difference in logits for the correct answer between the clean and corrupt runs. Specifically,

$$\text{Faithfulness} = \frac{\ell_{\text{patched}} - \ell_{\text{corrupt}}}{\ell_{\text{clean}} - \ell_{\text{corrupt}}} \tag{1}$$

where:

- ℓ_{clean} is the mean logit for the correct clean answer token from the clean runs,
- ℓ_{corrupt} is the mean logit for the correct clean answer token from the corrupt runs,
- ℓ_{patched} is the mean logit for the correct clean answer token from the patched runs.

This measures how well the patched activation restores clean behavior. A faithfulness score close to 1 indicates that patching restores the clean prediction, i.e., the patched position is not important for the model's prediction; while a low score indicates that the patched position is crucial for the model's prediction. Figures 9 and 10 show faithfulness scores across layers and positions, for attention modules and MLP layers. In both tasks, the last token of the prompt consistently emerges as the most influential, although in THR, early-layer MLPs also play a key role.

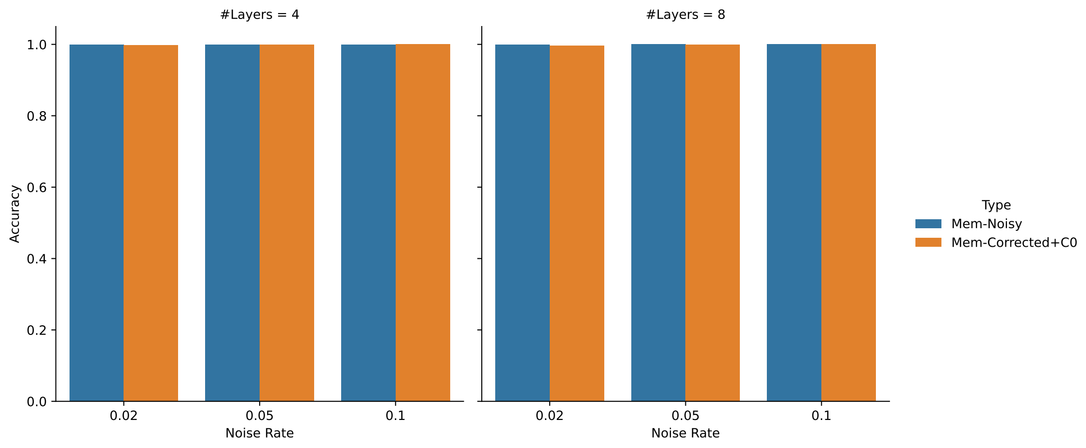
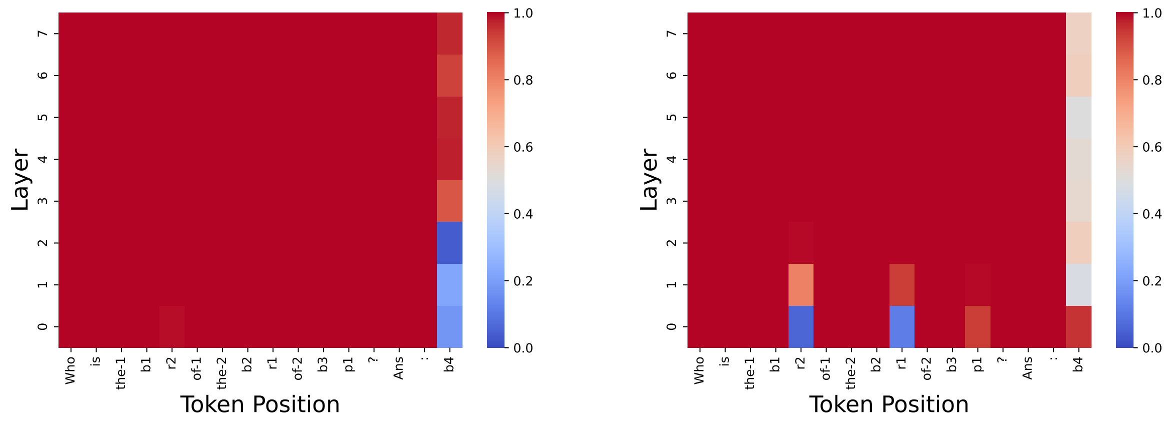


Figure 8: Accuracy on different prompts



(a) Attention output faithfulness

(b) MLP output faithfulness

Figure 9: Output faithfulness of attention and MLP layers on THR.

D Identifying generalization computation in hidden layers

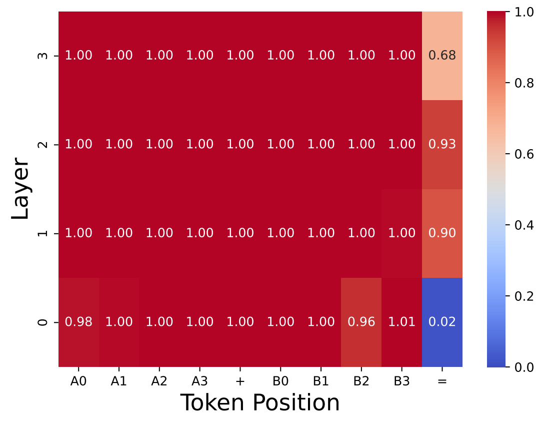
For both logit lens and linear probing, we use the residual stream of each layer at the final prompt token position, i.e., the = token in FDA and the blank token after the : token in THR, to predict the final answer token, i.e., `C0` for FDA and `P2` for THR. We experiment on three data splits: Mem-Noisy, Mem-Corrected, and Validation.

Because the models have memorized the noisy labels and can generalize to the clean labels of the validation set, the accuracy on both Mem-Noisy and Validation should be high, at least for the final layer. However, if $\mathcal{H}_1$ holds, that the models completely by pass generalization when producing memorized noisy labels, the accuracy on Mem-Corrected should be low, because the correct clean labels should not be detectable from the hidden representations of the noisy training instances.

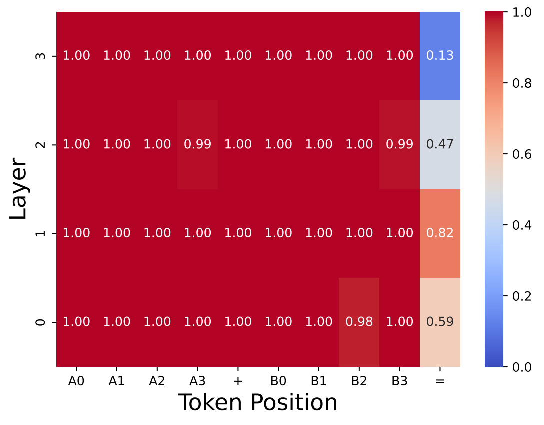
Linear probing We train linear probes, i.e., linear models with a single layer and without the bias term, to predict the final answer token from the residual stream of each layer. Specifically, we use 80% of the data from each split as the training data, and report the validation performance on the remaining 20% of the data. We train these linear probes for 200 epochs with a learning rate of 1e-3 and batch size 64, using Adam as the optimizer (Kingma and Ba, 2015).

Logit lens For FDA, besides linear probing, which follows the same setup as the linear probes for THR, we also experiment with logit lens. Specifically, we apply the model's final layer's layer normalization to the residual stream, and then decode the logits using the model's unembedding matrix. We then take the *token of the highest logit* as the predicted answer token.

The rationale for also including logit lens is that, a linear model can serve as a stronger baseline for



(a) Attention output faithfulness



(b) MLP output faithfulness

Figure 10: Output faithfulness of attention and MLP layers on FDA.

predicting `C0`. Concretely, if the linear model can memorize all the A0–B0 combinations, e.g., memorizing that it should output 8 if `A0` = 1 and `B0` = 7, it can achieve 50% accuracy on the validation set. If the model can further memorize all A1–B1 combinations, its accuracy can be even higher. However, this is not the case for logit lens, because it is training-free: it only detects information that is already present in the hidden representations. Moreover, we do not have this issue on THR, because the input-answer mappings in the validation set never appear in the training data.

E Circuit discovery: edge attribution patching

We use **edge attribution patching** (EAP; Syed et al., 2023) to identify the *circuits* responsible for generalization to unseen inputs (i.e., computing clean labels) and memorization of noisy labels.

The idea of EAP is similar to activation patching. Specifically, EAP builds on attribution patching (Nanda, 2023), an efficient approximation of activation patching (see §D). This approximation is based on a first-order Taylor expansion of the model's prediction: the change in the output caused by patching an intermediate activation z is estimated as the dot product between the change in z and the gradient of the output with respect to z on the corrupt run. Unlike activation patching, which requires three forward passes for each component of interest, attribution patching requires only two passes and a single gradient computation to estimate importance scores for all components simultaneously. This makes it significantly more efficient for large-scale analysis.

Instead of measuring the importance of modules, i.e., nodes in the computational graph, EAP measures the importance of edges, i.e., the connections between nodes, e.g., the influence from an attention head to a certain MLP layer. Similar to our experiments before, we estimate the circuit of generalization by using each prompt in the validation set as the clean prompt, and randomly sampling another validation prompt as the corrupt one. Similarly, to estimate the circuit of memorization, we use each prompt from the noisy training set as the clean prompt, and randomly sample another noisy training example as the corrupt one. We then similarly use *faithfulness* metric to quantify the importance of each edge.

We make two observations across different tasks and faithfulness thresholds for the obtained circuits (Figures 11 and 12). First, there are substantial overlaps between the circuits for generalization and memorization, indicating a tight coupling between the two mechanisms. Second, the generalization circuit is often a subset of the memorization circuit. This indicates that memorization mechanism is built on the top of the existing generalization mechanism, consistent with our observation that models first develop (a part of) their generalization mechanism, and only then start to memorize label noise (§3.1). This result is also consistent with our finding that the memorization mechanism relies on the generalization mechanism (§4.1). We also show the results for the sparsity-faithfulness tradeoff for both tasks for reference. Intriguingly, we observe that the memorization circuit on FDA takes a very large of edges: together with the milder effect we observe (§4.2) when ablating certain attention heads, this further suggests that memorization follows a distributed encoding across many different input tokens and intermediate results.

F Faithfulness computation for pre-memorization activation patching

Our activation patching experiments in §4.3 aim to study the influence of *MLP neuron activation changes* between the pre-memorization and post-memorization models on the model's predictions. We compute *faithfulness* for two data splits: Mem-Noisy and Validation: Mem-Noisy is used to study the influence of MLP activations on memorizing noisy labels, and Validation is used to study the influence of MLP activations on generalizing to clean labels. We also study two settings: pre-memorization patching and random patching. Similar to Appendix D, we use all instances from each data split as the clean prompts, and randomly sample another instance from the same split for each clean prompt as the corrupt prompt. This results in four sets of experiments, as illustrated in Figure 6b.

We follow the definition in Equation 1 to compute the faithfulness score (we restate the metric here for clarity):

$$\text{Faithfulness} = \frac{\ell_{\text{patched}} - \ell_{\text{corrupt}}}{\ell_{\text{clean}} - \ell_{\text{corrupt}}}. \qquad (2)$$

Specifically, in this experiment, given a clean–corrupt prompt pair, we compute the logits for the correct answer token of the clean prompt under three conditions:

- ℓ_{patched} is the mean logit from the patched run. In the *pre-memorization patching* setting, we use the same clean prompt but replace the MLP activation with that from the *pre-memorization model*. In the *random patching* setting, we use the same post-memorization model but replace the MLP activation with that from the *corrupt prompt*.

- ℓ_{clean} is the mean logit for the correct clean answer token when running the post-memorization model on the clean prompt.

- ℓ_{corrupt} is the mean logit for the clean prompt's answer token when running the clean prompt, but with **all MLP activations replaced** by those from the corrupt prompt. (the post-memorization model is also used here). We use this setup, instead of directly running the corrupt prompt, because it provides a lower bound on the influence of MLP activations alone, excluding changes in other parts of the model (i.e., we do not consider the contributions of attention modules here).

We obtain all values for estimating faithfulness by averaging across all prompt pairs in the corresponding data split.

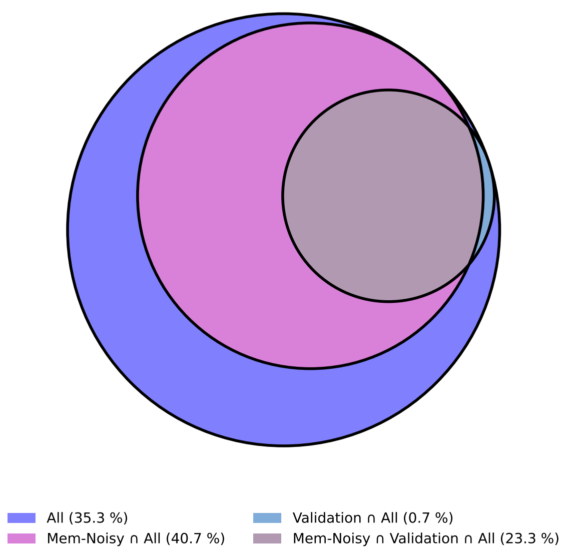
G Iterative Null-space Projection (INLP)

INLP is a popular method to remove linearly-encoded information from the hidden representations of a neural network (Ravfogel et al., 2020). Specifically, it iteratively perform the following two steps: (1) train a linear model to predict a target label from the hidden representations, e.g., the bridge entity in THR; and (2) project the hidden representations onto the null space of the linear model, so that the linear model cannot predict the target label anymore. This process is repeated multiple times, until it is no longer possible to train a linear model to predict the target label, i.e., the prediction accuracy is lower than a certain threshold ϵ. In our experiments, we set $\epsilon = 0.1$. In practice, we only need no more than three iterations to achieve this.

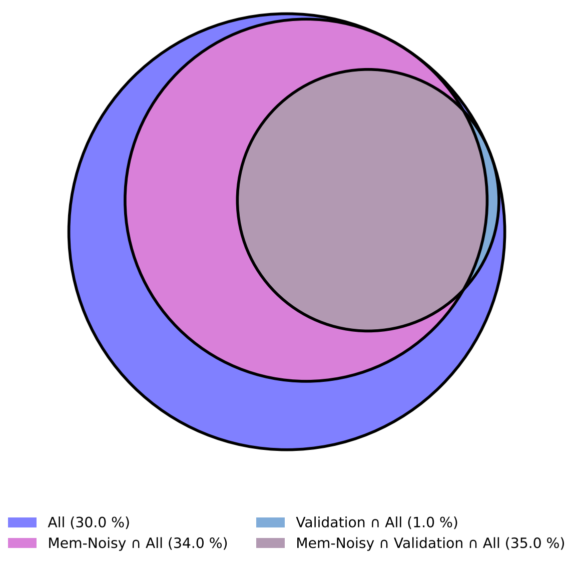
H Running Environment and AI usage

Environment We use a single NVIDIA A100 GPU with 80GB memory for our experiments. The main training and evaluation code is implemented in PyTorch (Paszke et al., 2019), and Hugging Face Transformers (Wolf et al., 2020), using Python 3.12.

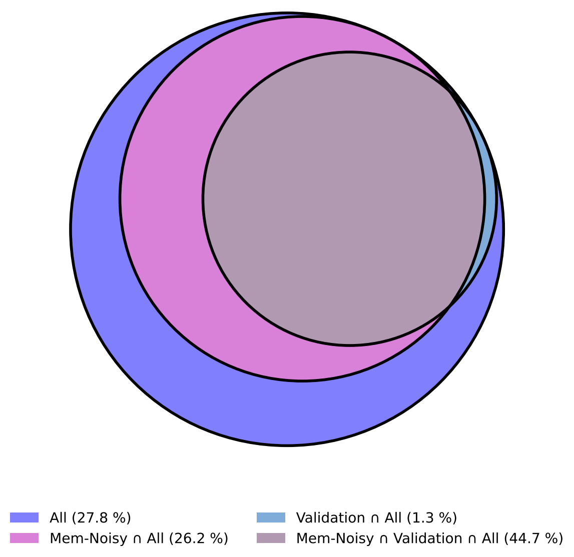
Use of AI assistants Our code is implemented with the help of ChatGPT, Google Gemini, and GitHub Copilot. We mainly use these tools to assist data visualization after obtaining the results. Moreover, the person entity names and relation names, as well as the templates for verbalizing the triplets into sentences and QA pairs, are also constructed with the help of ChatGPT. We also used ChatGPT to assist paper writing, in particular to revise and improve the clarity of the text.



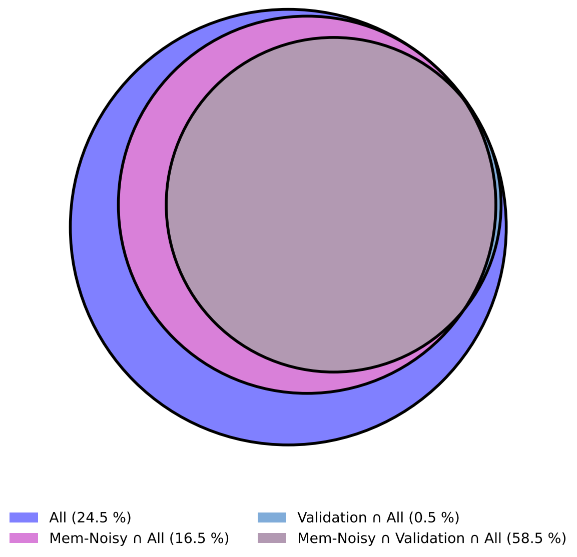
(a) 90%



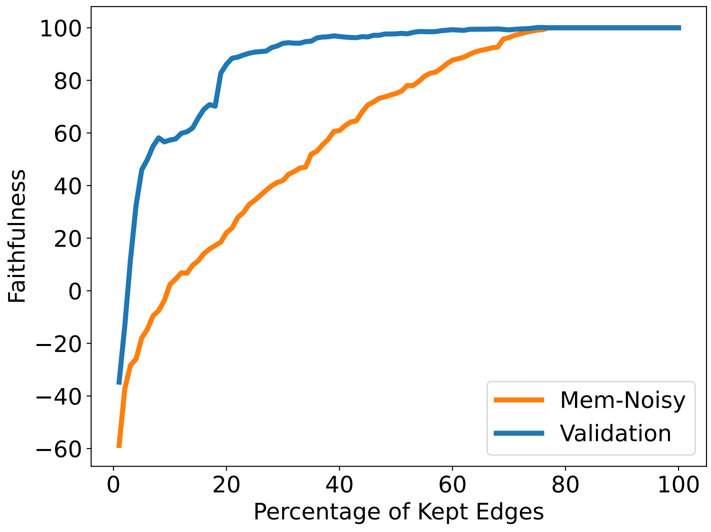
(b) 95%



(c) 97%

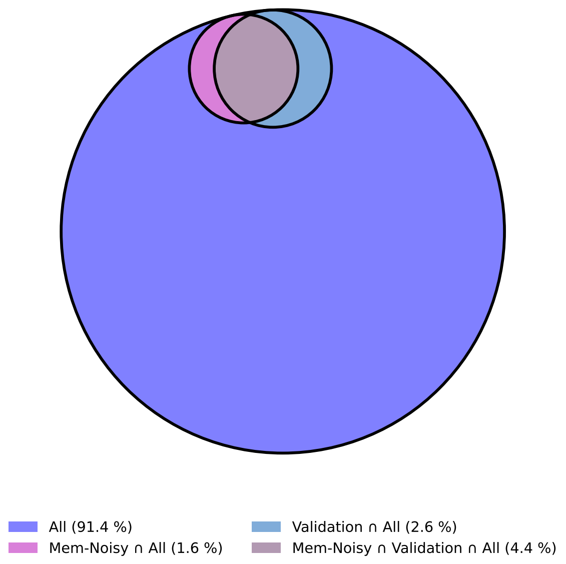


(d) 99%

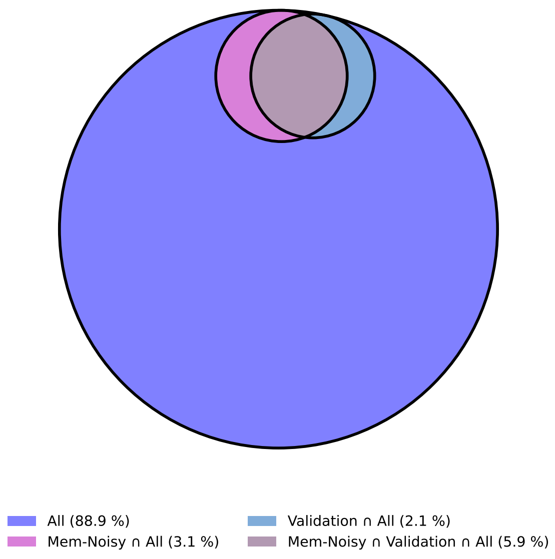


(e) Sparsity-faithfulness trade-off

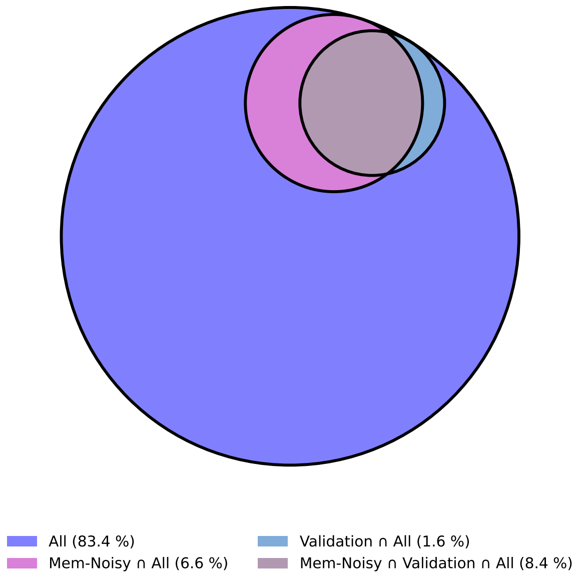
Figure 11: FDA circuit overlap across different faithfulness.



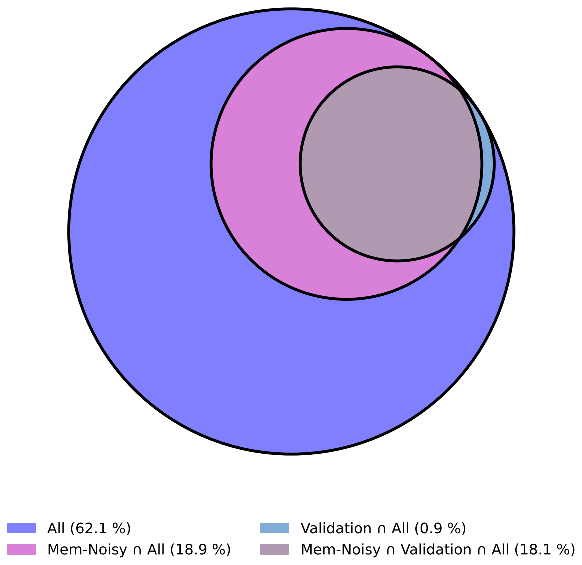
(a) 90%



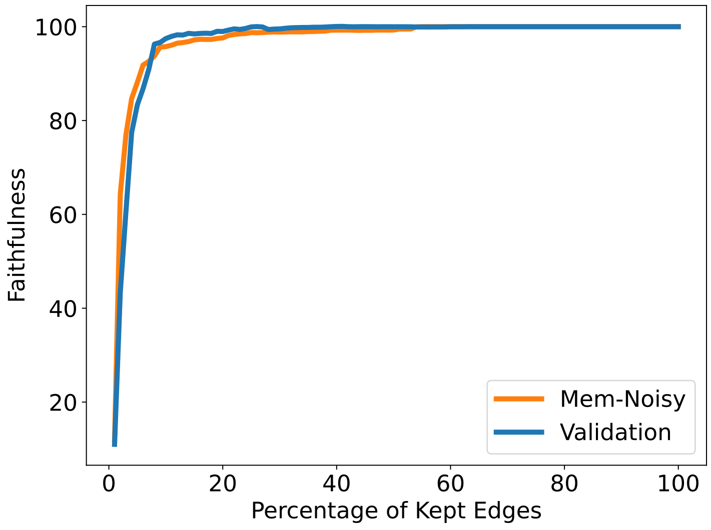
(b) 95%



(c) 97%



(d) 99%



(e) Sparsity-faithfulness trade-off

Figure 12: THR circuit overlap across different faithfulness.

I Additional results

Learning dynamics of THR We show the learning dynamics of THR in Figure 13, where we observe a similar trend as in FDA: the model learns to produce the generalizable reasoning output first on training instances of noisy labels which the model has never seen (although the performance never reaches 100%), but eventually memorizes the noisy labels. Also, in the generalization stage, the model's performance on Mem-Corrected closely matches that of Validation.

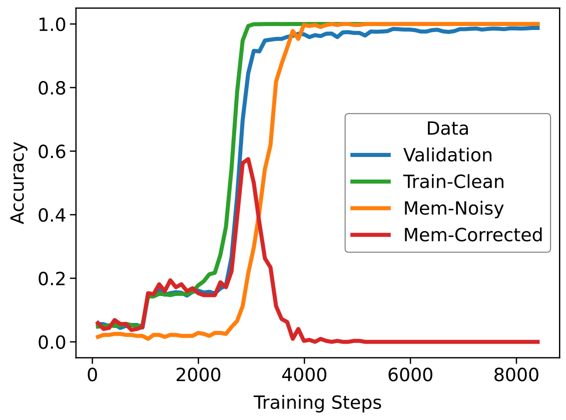


Figure 13: Learning dynamics of THR

I.1 Example outlier heuristics

We show more examples of outlier heuristics in Figure 14. Specifically, for each noisy training instance, we show the most influential neuron identified by the faithfulness drop by patching pre-memorization activation values. We consistently observe the outlier heuristics phenomenon.

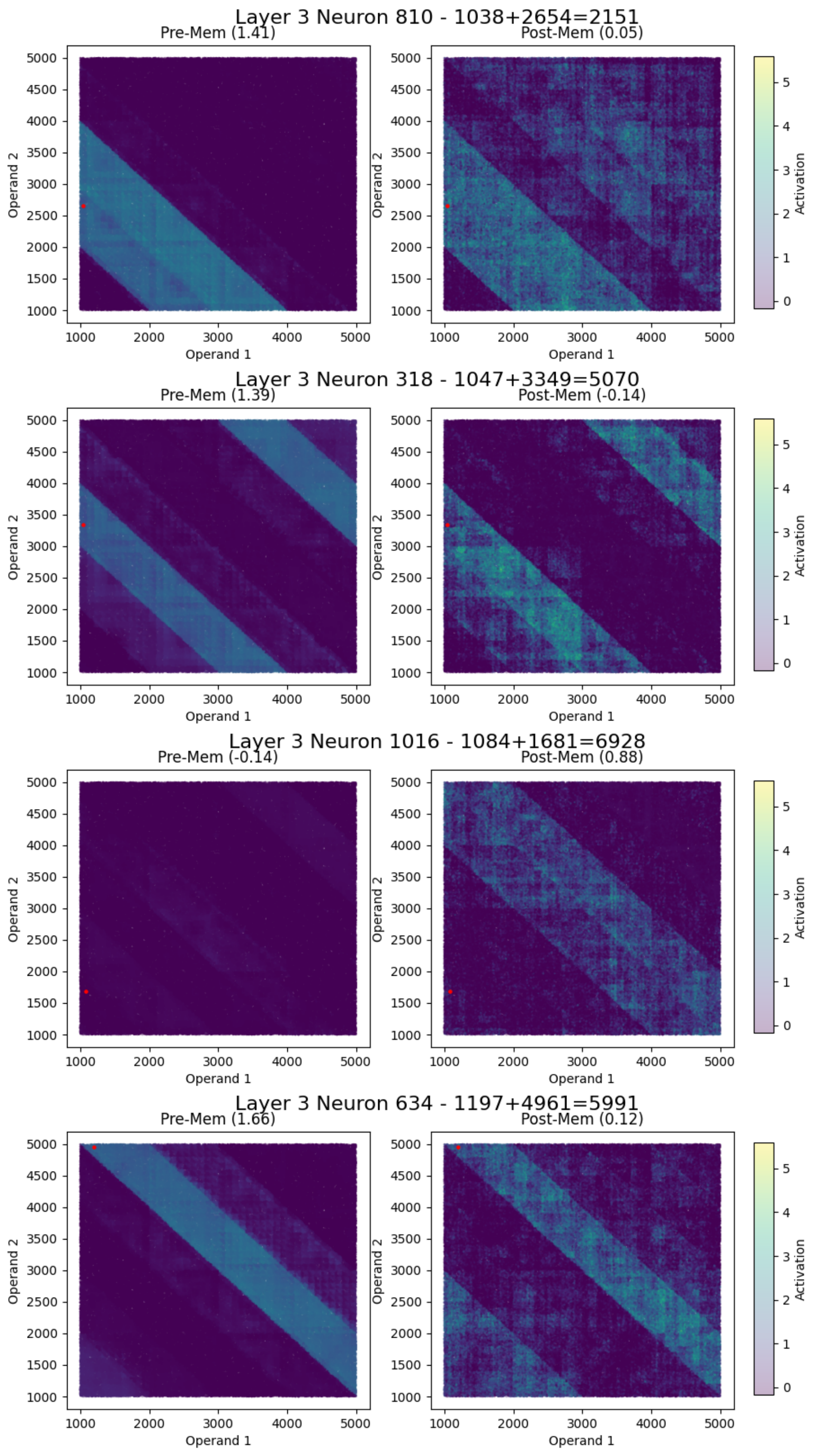


Figure 14: Examples of outlier heuristics: Activation patterns of the most influential neuron for four different noisy training instances. Each row shows the activation pattern for one instance. The red dot indicates the position of the noisy training instance.