DO TRANSFORMERS USE THEIR DEPTH ADAPTIVELY? EVIDENCE FROM A RELATIONAL REASONING TASK

Alicia Curth, Rachel Lawrence, Sushrut Karmalkar & Niranjani Prasad
Microsoft Research Cambridge
{aliciacurth,ralawrence,skarmalkar,niprasa}@microsoft.com

ABSTRACT

We investigate whether transformers use their depth adaptively across tasks of increasing difficulty. Using a controlled multi-hop relational reasoning task based on family stories, where difficulty is determined by the number of relationship hops that must be composed, we monitor (i) how predictions evolve across layers via early readouts (the logit lens) and (ii) how task-relevant information is integrated across tokens via causal patching. For pretrained models, we find some limited evidence for adaptive depth use: some larger models need fewer layers to arrive at plausible answers for easier tasks, and models generally use more layers to integrate information across tokens as chain length increases. For models finetuned on the task, we find clearer and more consistent evidence of adaptive depth use, with the effect being stronger for less constrained finetuning regimes that do not preserve general language modeling abilities.

1 INTRODUCTION

Large language models (LLMs) have made rapid advances over the last five years, with increasing model scale apparently driving the emergence of a priori unexpected capabilities (Brown et al., 2020; Wei et al., 2022; Ganguli et al., 2022). As models grow by stacking more layers, they gain additional computational *depth*, yet it has recently been called into question whether this depth is actually used efficiently: Csordás et al. (2025) find that the second half of a model's layers makes a much smaller contribution to the residual stream than earlier layers, and conclude through a series of empirical investigations that later layers are not performing fundamentally new computations.

These empirical findings stand in contrast with recent theoretical work emphasising depth as a critical bottleneck for transformers to solve algorithmic and reasoning tasks of increasing difficulty (Merrill et al., 2022; Sanford et al., 2024; Merrill & Sabharwal, 2025). The theoretical importance of depth has further motivated the insight that chain-of-thought approaches work because they equip models with additional effective computational depth (Merrill & Sabharwal, 2023), and has spurred investigations into recurrent transformer architectures that allow models to adaptively determine their own depth (Dehghani et al., 2018; Saunshi et al., 2025; Fan et al., 2024).

The tension between these perspectives raises a natural question: even if depth use appears inefficient on average, might models have learned to implicitly "reserve capacity" for harder tasks? Here, we thus define adaptive depth use as the property that the effective depth a model relies on to produce an output scales with task difficulty. While Csordás et al. (2025) and Hu et al. (2025) present some empirical evidence that this is not the case, we revisit this question with an empirical setup that affords us more precise control over how task difficulty is defined and how depth use is measured.

Outlook. We study a relational reasoning task, the family story relationship composition benchmark of Sinha et al. (2019), in which difficulty is controlled by varying the number of hops in the reasoning chain. We restrict answers to a single token, confining the computation of interest to a single forward pass. We apply logit lens (nostalgebraist, 2020) and causal patching (Zhang & Nanda, 2023) analyses to investigate how predictions and cross-token information integration, respectively, evolve differently across tasks of varying difficulty. Investigating the behavior of pretrained models, we consider five families of open-weights models ranging from 120M to 14B parameters (GPT-2, Pythia, Phi, Qwen2/2.5, and LLaMA), and find evidence that some larger models need fewer layers

to arrive at plausible predictions for easier tasks, and that models rely on more layers for information integration across tokens as chain length increases. Investigating models finetuned on the task (GPT-2 and Pythia across a range of sizes), the picture depends on the training regime: models trained under a more constrained setup (preserving general language modeling ability) do not arrive at plausible predictions earlier for easier tasks, but do use more of their earlier layers for information integration – effectively starting this process earlier in the network. Models trained under a less constrained regime show the strongest evidence of adaptive depth use: they use more layers overall for information integration, and arrive at plausible predictions earlier for easier tasks.

2 Background and related work

Setup: Transformers and logit lens. We consider only pre-Layer-Norm transformers with L layers. Their hidden states $h_l \in \mathbb{R}^{d\times T}$, where d is the hidden dimension and T is the sequence length, evolve across depth via residual connections as

$$h_l = h_{l-1} + f_l(h_{l-1}) = h_k + \textstyle\sum_{q=k}^{l-1} f_q(h_{q-1}) \tag{1}$$

where $f_l(h_{l-1})$ is a standard transformer block comprising attention, feed-forward, and layernorm operations. The predictive distribution over the next token at position t is computed from the final hidden state $h_{L,t} \in \mathbb{R}^d$ via the language model head, parameterized by the unembedding matrix $W_U \in \mathbb{R}^{V\times d}$, a final layernorm LN_f, and a softmax:

$$p_t = \text{softmax}(\text{logit}_t) \text{ with } \text{logit}_t = W_U LN_f(h_{L,t}) = W_U LN_f\Big(h_{l,t} + \textstyle\sum_{k=l}^{L}[f_k(h_{k-1})]_t\Big) \tag{2}$$

The logit lens (nostalgebraist, 2020; Geva et al., 2022) is an interpretability technique that exploits the residual structure in Eq. (1) to inspect intermediate hidden states by projecting them into vocabulary space using the language model head, effectively assuming that the remaining layer contributions $\sum_{k=l}^{L}[f_k(h_{k-1})]_t = 0$. The resulting early read-out distribution at layer l is:

$$\tilde{p}_{l,T} = \text{softmax}(W_U LN_f(h_{l,T})) \tag{3}$$

This method has been widely used to interpret language model hidden states in vocabulary space, e.g. Halawi et al. (2023) use it to show that accuracy on correct answers peaks at intermediate layers before declining when models are provided with misleading in-context examples, Merullo et al. (2024) use it to reveal distinct stages of processing in factual recall tasks, and Lepori et al. (2025) use it to show that final answers are decodable halfway through the network in tasks requiring significant contextualisation (e.g. disambiguating whether "bank" refers to a financial or geographical entity). A recent *trained* alternative to the logit lens is the tuned lens (Belrose et al., 2023), which fits an affine probe at each layer to predict the final model output, compensating for future layer transformations. We choose to rely on the logit lens here because we are interested in how close the residual stream is at each layer to a state from which the answer can be directly read out, rather than what it would become after further processing.

Related work: How do transformers use their depth? Understanding what precisely happens at different depths in language models has recently received increased attention. Motivated in part by observations that some layers can be shuffled or removed without significantly degrading performance (Gromov et al., 2024; Sun et al., 2025), Lad et al. (2024) conduct an empirical investigation across model families and identify four distinct phases of inference with increasing depth in LLMs : detokenization, feature engineering, prediction ensembling (as features are aggregated into candidate next-token predictions), and residual sharpening (as irrelevant features are suppressed to finalize the prediction). Similarly, Queipo-de Llano et al. (2025) identify three stages: an initial mixing stage that builds context, a subsequent compression stage, and a final refinement stage that constructs task-specific representations. Gupta et al. (2025) posit that earlier layers act as statistical guessers that steer models toward high-probability tokens, while later layers refine predictions by incorporating context.

Most related to our work are two recent papers that study depth use directly. Csordás et al. (2025) question whether LLMs use their depth efficiently, concluding through a series of empirical investigations that deeper models are not using their additional layers for more involved computation. They

2-hop example

Michael has a son called John.
John has a sister named Elizabeth.
Therefore, Elizabeth is Michael's daughter

5-hop example

Stella is the wife of Kenneth.
Gregory is Stella's son.
Gregory has a son called Jim.
Adrian is a brother of Jim.
Juan is Adrian's brother.
Therefore, Juan is Kenneth's grandson

Figure 1: **Example CLUTRR stories** for 2-hop and 5-hop reasoning. Each sentence states a family relation between two people. The model must compose the chain of relations to infer the target relationship between the query pair.

show that layers in the second half of the model contribute substantially less to the residual stream than those in the first half and that skipping later layers has a smaller effect on predictions. They also find no evidence that models use later layers to compose subresults when multi-hop reasoning is required, using pre-defined difficulty levels (MATH dataset) and different hop numbers (MQuAKE dataset) and averaging depth scores across all tokens in the generated answers. Hu et al. (2025) build on these findings and study potential drivers of effective depth, finding no systematic difference in relative depth usage across model sizes or task difficulty, using metrics that track aggregate changes in the residual stream averaged across all answer tokens in HellaSwag (natural language understanding), GSM8K (grade school math), and AIME24 (high school math contests). We consider a more controlled setup that allows us to vary difficulty more systematically, restricting evaluation to single-token answers, and find evidence that partially diverges from these conclusions.

3 Empirical investigation: Do LLMs use their depth adaptively in a multi-hop relational reasoning task?

3.1 Experimental setup: Task and analyses

The CLUTRR task. We evaluate relational reasoning using the Compositional Language Understanding and Text-based Relational Reasoning (CLUTRR) data generator[1] of Sinha et al. (2019), which generates k-hop family stories requiring the composition of a chain of k family relations. We use the simplest version of the generator, which produces stories free of distracting facts or language, ensuring that task difficulty stems solely from resolving family relationships of varying chain length. We use the default generator settings, which create a three-level family tree (three generations) with at most 4 siblings, to generate 2 to 10-hop stories for the main analyses in Section 3.2; for the longer stories in Section 3.3, we need to expand the maximum number of children to 6. With this setup, the possible answers are the tokens $\mathcal{F}$ = `[mother, father, grandfather, grandmother, son, daughter, grandson, granddaughter, brother, sister, uncle, aunt, nephew, niece]`[2]. We modify the generator so that all stories end with the sentence *"Therefore, PersonA is PersonB's"*, framing the reasoning task as a next-token prediction problem and allowing us to focus on computation at a single token. Examples of generated stories are shown in Fig. 1. We generate 100 test examples per k-hop level.

Logit lens analyses of $h_{l,T}$. Part of our investigation centers on applying the logit lens to decode and analyse the model's hidden states. Specifically, we focus on the probability distribution $\tilde{p}_{l,T}$ obtained by projecting the hidden state at the final token position T (the "'s" token) through the language modeling head, as defined in Eq. (3). We begin by verifying that analysing hidden states in this way yields meaningful signal, by monitoring the total probability mass assigned to the set of permitted family relation tokens at each layer: $p_l^{\text{fam}} = \sum_{w \in \mathcal{F}} \tilde{p}_{l,T}(w)$. We then monitor whether the correct answer is identified at each layer via two metrics: (i) *correctness*, whether the token assigned the highest overall probability ($\arg\max_{w \in \mathcal{V}} \tilde{p}_{l,T}(w)$) is the correct answer; and (ii) *constrained correctness*, whether the top-ranked token among the relation tokens ($\arg\max_{w \in \mathcal{F}} \tilde{p}_{l,T}(w)$) is the

[1] Adapted from `https://github.com/facebookresearch/clutrr/`.

[2] To be able to focus on computation at a single tokens, we restrict our analysis to questions with single-token answers: this is the case for all required relations in $\mathcal{F}$ but excludes the in-law relations that were part of the original generator as they decode to multiple tokens in most models.

correct answer. In addition to monitoring h_l through the logit lens, we replicate part of the h_l analyses in Csordás et al. (2025); Hu et al. (2025) by tracking the relative contribution of each layer $\Delta_l = h_l - h_{l-1}$ to the residual stream $\frac{||\Delta_l||_2}{||h_{l-1}||_2}$ and the similarity of the update to the residual stream $cossim(\Delta_l, h_{l-1})$ in Appendix B.6.

Causal patching analyses. To better understand where and how information is processed across network depth, we additionally conduct *causal patching experiments* (Meng et al., 2022) with counterfactual relationship replacements, inspired by the setup that Wu et al. (2025) use to study how transformers learn variable binding. Given a test story, we replace a single relationship token $t^r = a$ with a counterfactual relation of the same gender $t^r = b$, changing also the ground truth query relationship. We compute the full hidden states associated with original prompt h_l^a and mutated prompt h_l^b. A causal patching experiment then proceeds by examining the effect of intervening on the hidden state $h^b_{l^*,i}$ of the modified story at a single layer l^* and a single token index i by replacing it with the corresponding original $h^a_{l,i}$ and then running the model forward from that point. That is,

$$h_{l,j}^{patch_{l^*,i}:b\to a} = \begin{cases} h^b_{l,j} & \text{if } l < l^* \text{ or } j < i \text{ or } (l = l^* \text{ and } j \neq i) \\ h^a_{l,i} & \text{if } i = j \text{ and } l = l^* \\ \left[h_{l-1}^{patch_{l^*,i}:b\to a} + f_l\left(h_{l-1}^{patch_{l^*,i}:b\to a}\right)\right]_j & \text{if } l > l^* \text{ and } j \geq i \end{cases}$$

We track whether this intervention recovers the original model prediction at the final token T using a normalized logit difference of the original predicted token o and the new prediction c similar to Zhang & Nanda (2023); Wu et al. (2025) as

$$Rec^{patch_{l^*,i}:b\to a} = \max\{0, \frac{\text{ld}_T^{patch_{l^*,i}:b\to a}(o,c) - \text{ld}_T^b(o,c)}{\text{ld}_T^a(o,c) - \text{ld}_T^b(o,c)}\} \text{ where } \text{ld}^{(\cdot)} = \text{logit}_T^{(\cdot)}[o] - \text{logit}_T^{(\cdot)}[c].$$

Intuitively, this allows us to track how information moves across sequence positions and depth. To see this, note that patching the embedding ($l = 0$) at the modified token t^r would restore the prediction perfectly, as would patching the final-layer hidden state at the final token T. Being able to recover the original prediction by patching at layer $l^* > 0$ at the modified token ($i = t^r$) then means that the semantic information in this token has either not yet propagated to future tokens, or is retrieved by the final token T directly from position t^r. Conversely, recovering the original prediction by patching at the final output token ($i = T$) before the last layer ($l^* < L$) implies that all information about the modified token has already been integrated at the final position, with no future layers integrating further conflicting information about t^r from non-patched positions. Being able to recover the original prediction by patching hidden states at token i between t^r and T and layer l indicates that information about t^r flows to T via i at layer l.

Note that the recovery score is meaningful only when the model changes its prediction in response to the counterfactual (which is the correct behavior, since the true answer changes). We therefore sample only stories for which the model's top prediction changes under the counterfactual, regardless of whether either prediction is correct. To ensure that all counterfactual stories have valid single-token answers, we start with simple stories containing only brother/sister relationships (correct answer: brother/sister) and intervene by replacing them with father/mother relationships at intermediate tokens, or uncle/aunt relationships at the first and last tokens (correct answer in both cases: uncle/aunt). Due to the high computational cost of patching analyses – which require $k \times T \times L$ forward passes per example – we restrict the analysis to a subset of models, hop counts, and examples throughout, using $n = 30$ examples per configuration when available[3]. In Appendix B.5, we repeat the analyses for the original stories with more involved relationship chains and instead replace existing relations with brother/sister regardless of whether this leads to valid target answers and find consistent trends.

3.2 Do pretrained transformers use their depth adaptively?

This section analyses how pretrained models use their depth on the family relations task. We consider five open-weights model families[4] (GPT-2, Pythia, Phi, Qwen2, Qwen2.5, and LLaMA-3),

[3]For some model–hop combinations, fewer than 30 examples flip their prediction upon the intervention.

[4]Note that we use the stories as prompts exactly as presented in Fig. 1 in order to constrain computation to a single token; we are therefore *not* using the chat template that some of these models are post-trained to expect.

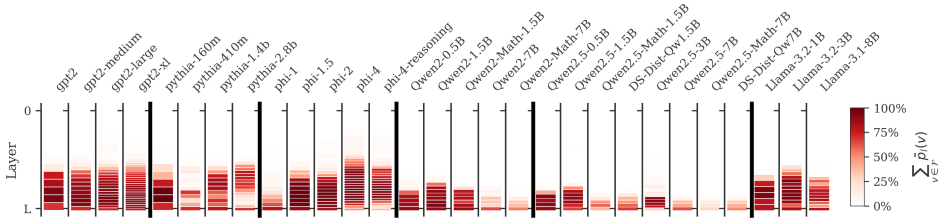


Figure 2: **Probability assigned to family relation tokens** when decoding the hidden states of the final token using the language modeling head directly, by layer across different pretrained model sizes and families, averaged across hops. Family relations are decodable at layer $l < L$ across all models.

spanning a range of sizes up to 14B parameters. The upper end of this range is determined by the computational cost and memory footprint of the analyses; within this constraint, we select sizes that allow us to study trends within each family. This gives us a diverse set of architectures and training recipes while keeping the analyses tractable.

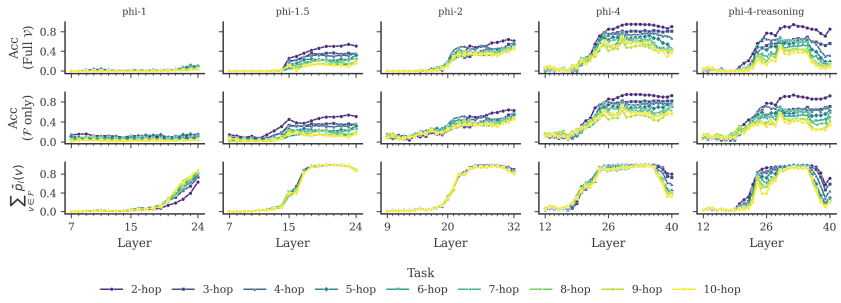
• **Observation 1:** ***Family relations are decodable from around two-thirds depth in all models; final layers appear to diffuse predictions.*** We begin by confirming whether answers can be decoded from early hidden states using the language model head directly, as shown in Fig. 2. Strikingly, in all models there is a layer around two-thirds of the way through the network from which the logit lens decodes into family relationships with very high probability. This aligns precisely with the observation in Lad et al. (2024) that the early layers are responsible for detokenization and feature engineering: in our setting, after approximately two-thirds of the layers, the model has arrived at a region of the hidden space that is semantically meaningful for the task. The precise depth at which relations become decodable varies across model families but is consistent within them. Qwen2.5 models are decodable the latest, while Pythia models exhibit non-monotonic patterns.

Phi-4, by contrast, is decodable at the earliest point. We also observe that the probability mass assigned to family tokens dips in the final layers across almost all models. As shown in Fig. B1 in the appendix, the entropy of the induced distribution first decreases within the decodable layers, then rises again in the final layers[5]. This is interesting because it suggests that the final layers are not simply refining the predicted answer but also restructuring the distribution over answers, perhaps to achieve better calibration. A similar effect is observed by Lv et al. (2024) in the context of factual recall, who conclude that the final layer has an "anti-overconfidence" effect.

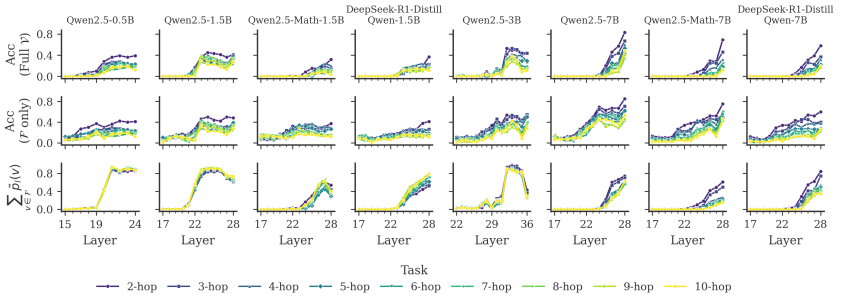
• **Observation 2:** ***Larger models need less depth to arrive at plausible answers for easier tasks; smaller models show little such differentiation.*** We now turn to the core question of whether pretrained models use more layers to arrive at correct answers for harder tasks, by examining decoded answer accuracy and family token probability per layer and per task. We observe that the answer to this may depend on model size: In Fig. 3, we consider models in the Phi and Qwen2.5 families (results for other model families are included in Section B.1). For most smaller models, accuracy and family token probability begin to rise at the same layer across all tasks: while they reach different absolute levels, they do so at similar rates, and absolute performance is poor across the board. For the largest models, a somewhat different pattern emerges: in Phi-4-Reasoning, the Qwen2.5-7B and Qwen2-7B models and Llama-3.1-8B, top-token accuracy and family token probability for easier tasks begin to rise several layers before they do for harder tasks, suggesting that these models may indeed need more depth to process stories with longer relationship chains.

• **Observation 3:** ***Longer relationship chains generally induce earlier cross-token information integration.*** We complement the logit lens analyses, which investigate when information becomes decodable across depth, with causal patching experiments, which provide insight into when information is propagated to and integrated at the final token. In Fig. 4, we focus on patching at (i) the replacement token t^r and (ii) final token T, to investigate whether contextual information about t^r is (i) processed and (ii) integrated into the final answer at different depths for tasks of different difficulty. We present examples of full patching trajectories in Section B.2. Across the considered models, we find that information gets processed at earlier layers at the replacement token t^r for

[5]This is particularly striking for the Phi-4-Reasoning model, where we observe up to 50% accuracy on the 10-hop task at intermediate layers, dropping back to 10% at the final layer, where the top token is often "___", particularly for longer stories (see e.g. the example in Fig. B3). We believe this may be partly attributable to this model being post-trained to expect certain formats, which we are not using.

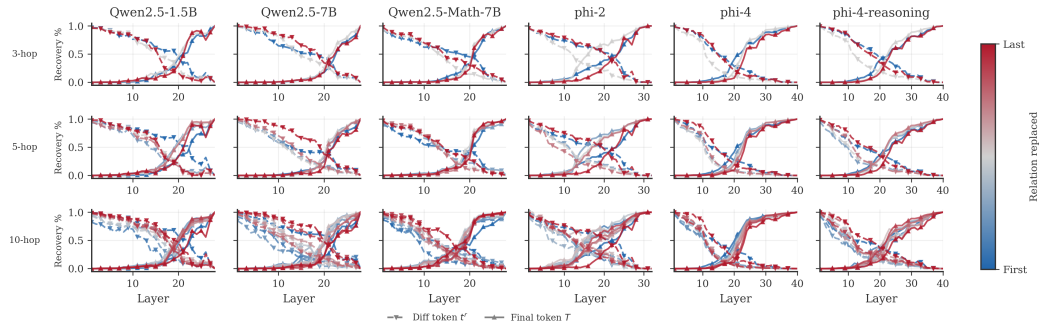


(a) Phi Family

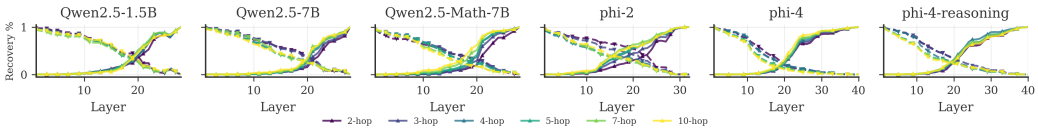


(b) Qwen2.5 Family

Figure 3: **Logit lens results for Phi and Qwen2.5 families.** Correctness, constrained correctness and probability assigned to family relation tokens by logit lens predictions by layer, colored by hops. X-axes (layers) are left-truncated for better readability; previous accuracies & p_l^{fam} are zero.



(a) Average recovery score by replaced relation across different hops (rows), colored by relation replaced.



(b) Average recovery score across all relation replacement positions, colored by number of hops.

Figure 4: **Causal patching results.** Average recovery score at the replaced token t^r (dashed lines) and the final token T (solid lines) by model depth.

longer stories (especially relationship tokens in the middle of longer stories in panel (a)) as indicated by recovery scores dropping at earlier layers, meaning that information is mixed into the residual streams of future tokens sooner for longer stories[6]. Interestingly, recovery scores at the final token T also begin to rise earlier on average for longer stories (panel (b)), suggesting that information is integrated at the final position earlier, potentially allowing more layers for subsequent answer refinement. While the former pattern (earlier mixing at t^r) appears also in the stories with more relationships in Fig. B14, the latter (earlier integration at T) does not. Beyond these trends, there appear to be limited consistent patterns to the order in which information about individual relations is integrated across models, though most models integrate information about the last relationship

[6]An exception to this rule is the pretrained gpt2-large in Fig. 7 (b, left), which does not appear to integrate information earlier at t^r for longer chains but instead integrates information later at T.

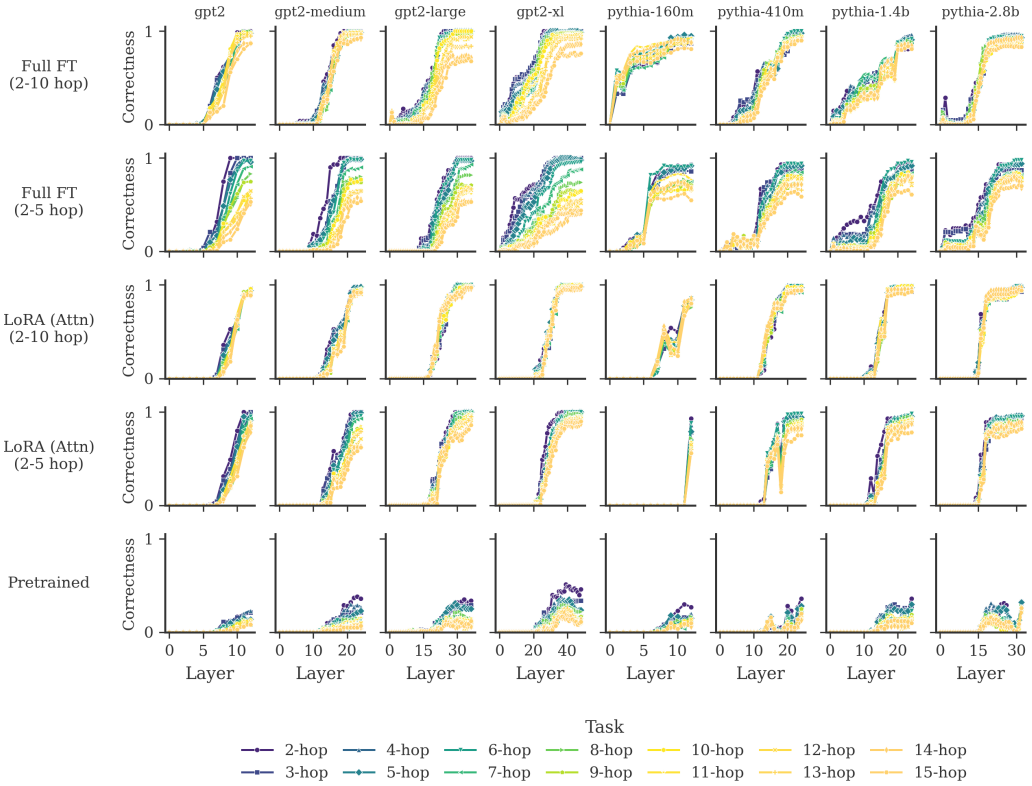


Figure 5: **Accuracy of the top logit lens prediction** across models (columns) and training regimes (rows) by layer and colored by number of hops.

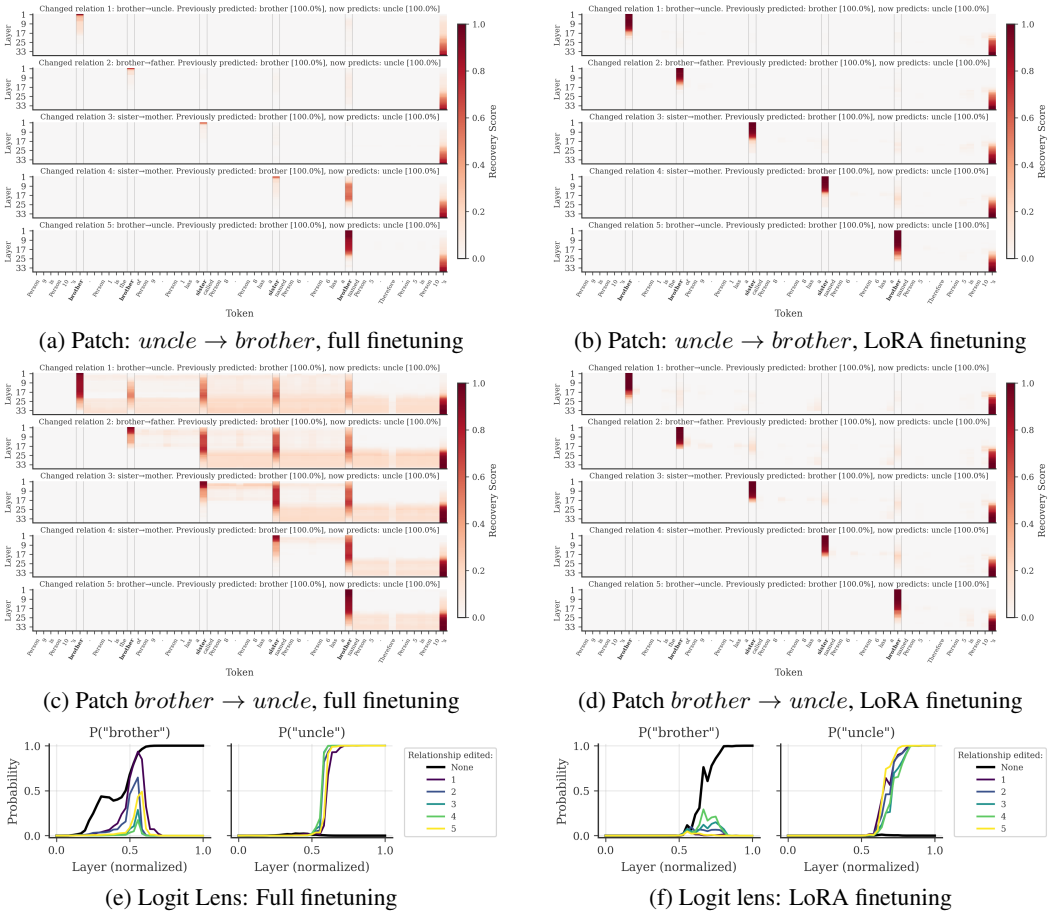
last. Qwen2.5 models appear to fully integrate information about the first relationship at T later than Phi models for longer stories.

Overall, the results in this section indicate that pretrained models do show some variation in depth use across task difficulty: larger models appear to need fewer layers to arrive at plausible answers for easier stories, and models generally use more layers to integrate information across tokens as chain length increases. This finding stands in partial contrast to Hu et al. (2025). We hypothesize that their setup was not sufficiently controlled to detect these differences: task difficulty was more loosely defined, and their metrics were averaged over many tokens. Moreover, the representation similarity metrics they employ (see Appendix B.6) are harder to interpret than the logit lens and causal patching analyses we conduct here, which are made possible precisely by the controlled nature of the task and the restriction to single-token answers.

3.3 Do transformers finetuned to solve the task use their depth adaptively?

Finetuning setup. Next, we finetune the GPT-2 and Pythia model families to solve the task, allowing us to investigate whether models specialised for the task *learn* to use their depth adaptively. We consider two finetuning regimes: (i) LoRA finetuning (Hu et al., 2022) of only parameter matrices in the attention modules, and (ii) full finetuning of the whole model. In both cases, we supervise using the loss on the answer token only (completion-only supervision). We train on 1000 examples consisting of an equal mixture of (i) 2–5-hop and (ii) 2–10-hop stories for 20 epochs. Unless indicated otherwise, we present results for models finetuned on 2–10-hop stories by default.

Our test set evaluates generalisation in two ways. First, within-task generalisation is assessed using the train–test split functionality of the CLUTRR generator, which ensures that the specific combinations of relations in the test set were not seen during training. Second, to test length generalisation, we evaluate on 11–15-hop stories, for which we need to increase the maximum number of allowed siblings to 6. Note that, as the CLUTRR generator is restricted to three-generation family trees, longer stories may contain structural shortcuts and necessarily include a higher proportion of sibling relationships. Finally, since family trees are shared across examples in the CLUTRR generator



(a) Patch: $uncle \rightarrow brother$, full finetuning

(b) Patch: $uncle \rightarrow brother$, LoRA finetuning

(c) Patch $brother \rightarrow uncle$, full finetuning

(d) Patch $brother \rightarrow uncle$, LoRA finetuning

(e) Logit Lens: Full finetuning

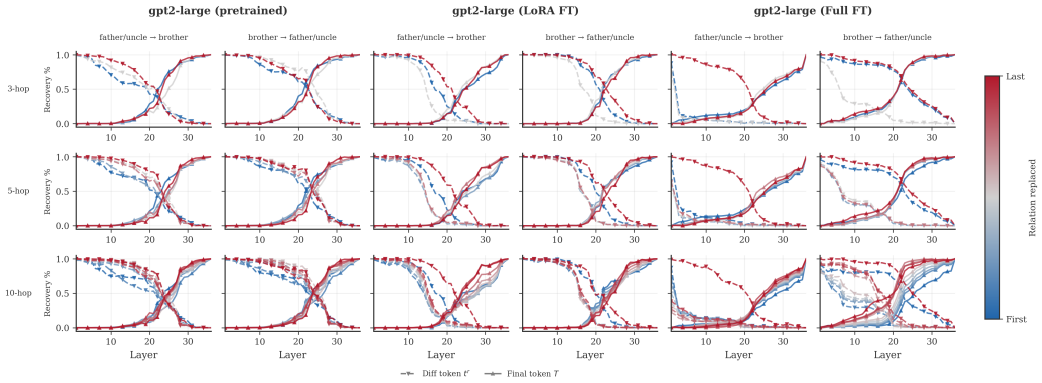
(f) Logit lens: LoRA finetuning

Figure 6: **Full causal patching and logit-lens trajectories** for a 5-hop example of finetuned GPT2-large versions.

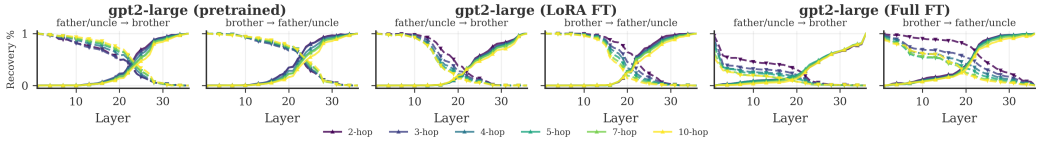
(making it possible in principle to solve the original task by memorising relations by name rather than reasoning about them) we rename all individuals to Person1 through PersonK and randomise the order in which numerical indices appear. Further implementation details are provided in Appendix A.

• **Observation 4:** ***All models learn the task, but only LoRA finetuned models length-generalise and preserve general language modeling ability; full finetuning causes earlier decodability but does not length generalise.*** We begin by considering model performance and make a number of interesting observations. Examining correctness in Fig. 5, all models regardless of size and finetuning regime successfully learn to solve stories of in-domain length not seen during training. Second, LoRA finetuning yields models that length-generalise to longer hop counts without performance loss, while full finetuning results in degraded performance on longer stories. Third, and perhaps unsurprisingly, restricting parameter updates to the attention heads via LoRA leaves the semantic structure of the residual stream largely unchanged: family relations remain decodable from the language modeling head at a similar depth as in the base models. Full finetuning, by contrast, causes family relations to become decodable much earlier, particularly in larger models (see also Fig. B6 in the Appendix). It is worth noting that aggressively finetuning on the final answer token only causes the fully finetuned models to lose their general language modeling ability entirely, while LoRA finetuned models retain it; see Table 2 in Appendix B.7. This observation is in line with Biderman et al. (2024) who find that LoRA learns *and* forgets substantially less than full finetuning.

• **Observation 5:** ***Fully finetuned models show evidence for adaptive depth use in logit lens decoding; LoRA finetuned models do not.*** Turning to the key question of whether models learn to use their depth adaptively, the logit lens results in Fig. 5 indicate a clear split between the two regimes. The LoRA finetuned models remain closely aligned with their base model counterparts, solving tasks at similar depths across all hop counts, providing little evidence of depth use adapting to task



(a) Average recovery score by replaced relation across different hops (rows), colored by relation replaced.



(b) Average recovery score across all relation replacement positions, colored by number of hops.

Figure 7: **Causal patching results for GPT2-large**, pretrained and finetuned. Average recovery score at the replaced token t^r (dashed lines) and the final token T (solid lines) by model depth.

difficulty. The fully finetuned models tell a different story: accuracy for harder tasks rises later in the network, especially for larger models, suggesting that these models do learn to process the task more iteratively with depth. This may also explain why fully finetuned models fail to length-generalise: if a model has learned to use its full depth for the hop counts seen during training, it has no capacity in reserve for longer stories.

• **Observation 6:** ***All finetuned models learn to speculatively integrate relationship information across tokens, with fully finetuned models devoting more of their total depth to this.*** We further investigate depth use via causal patching. Unlike pretrained models, which were not trained on the task and are unlikely to anticipate the final query, finetuned models could in principle learn to use the hidden representations of all tokens $t < T$ to speculatively resolve relationship compositions as they appear in the text. This is precisely what we observe in Fig. 6 (see Appendix B.4 for additional hop counts): information about earlier relationships appears to be integrated into the residual streams of subsequent family tokens, as well as some intermediate tokens. The fully finetuned models learn to use the residual stream of all tokens at all depths for this purpose, while the LoRA finetuned models retain the detokenization phase of the base model, during which no semantic information is mixed across tokens. Recovery scores at intermediate family tokens are larger for the fully finetuned models than for the LoRA models, indicating that more information is routed via intermediate tokens. Strikingly, patching the hidden states of the modified story from uncle back to brother is much less successful than patching in the opposite direction. In Fig. B9 in the Appendix we show that this is because the models have learned to attend differently (generally less) to brother/sister relations—likely because these are easier to resolve[7]—while uncle/father relations receive substantially more attention at future layers and tokens. Patching a single hidden state from uncle to brother therefore has little effect, since information about the harder relationship has already propagated forward; the reverse patch is far more effective precisely because limited information about the easier relation would have been passed forward in the first place.

• **Observation 7:** ***Both finetuning regimes use more depth for harder tasks, but fully finetuned models use substantially more depth overall for cross-token mixing and show clearer ordering of information integration.*** In Fig. 7(a), we aggregate across more patching trajectories and confirm that these patterns generalise (with results for other models in Appendix B.4). The LoRA finetuned models use approximately the same range of layers as the pretrained models to mix information

[7]In most cases, adding a sibling relationship to a chain does not change the final answer, whereas adding a parent relationship will.

across tokens, but change the order in which information is integrated, incorporating information about the first relationship substantially later. The fully finetuned models use much more of their depth for cross-token mixing (i.e. layers where recovery scores at both t^r and T are low), and devote more depth to processing some relationships (father/uncle) than others (siblings). We also find that the fully finetuned models most clearly integrate (i) information about the first relation last and (ii) information about the last relation first into the final prediction, while in the LoRA case this separation is less pronounced. In Fig. 7(b), we examine how information integration varies with task difficulty. Both the LoRA and fully finetuned models learn to use more of their depth for harder tasks, mixing information earlier and completing integration later for longer chains.

4 Conclusion and Discussion

In this work, we provide empirical evidence that transformers can adapt their use of depth to task difficulty, using a controlled multi-hop relational reasoning task based on family stories. Investigating pretrained models, we find that some models use fewer layers to arrive at plausible answers for easier stories, and that models generally use more of their depth to integrate information across tokens for longer relationship chains. Investigating finetuned models, we find that models trained directly on the task use more layers to integrate information for harder tasks, and alter the order in which information is integrated relative to the base model. Models finetuned more aggressively, without regard for preserving general language modeling ability, use substantially more of their total depth for the task compared to those finetuned with LoRA on attention parameters only, and show clearer evidence of arriving at plausible answers earlier for easier tasks.

While our results provide some evidence of adaptive depth use in pretrained models, our finetuning experiments also highlight that the need to preserve general language modeling ability likely constrains which layers are available to perform additional computation for harder reasoning tasks (e.g. needing to reserve depth for the apparent detokenization function of earlier layers). At the same time, our decodability results reveal that different model families arrive at a semantically meaningful region of the residual stream (one that is directly decodable via the language model head) at markedly different depths. Since these models are architecturally similar, this suggests that differences in training give rise to different pressures for the residual stream to remain aligned (or become misaligned) with the output embeddings. What drives this (mis)alignment, and whether the proportion of the network devoted to each stage of inference is steerable, are interesting questions for future work.

References

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, et al. Lora learns less and forgets less. *arXiv preprint arXiv:2405.09673*, 2024.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Róbert Csordás, Christopher D Manning, and Christopher Potts. Do language models use their depth efficiently? *arXiv preprint arXiv:2505.13898*, 2025.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. *arXiv preprint arXiv:2409.15647*, 2024.

Jaden Fiotto-Kaufman, Alexander R Loftus, Eric Todd, Jannik Brinkmann, Caden Juang, Koyena Pal, Can Rager, Aaron Mueller, Samuel Marks, Arnab Sen Sharma, et al. NNsight and NDIF: Democratizing access to foundation model internals. *arXiv preprint arXiv:2407.14561*, 2024.

Deep Ganguli, Danny Hernandez, Liane Lovitt, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova Dassarma, Dawn Drain, Nelson Elhage, et al. Predictability and surprise in large generative models. In *Proceedings of the 2022 ACM conference on fairness, accountability, and transparency*, pp. 1747–1764, 2022.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pp. 30–45, 2022.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. The unreasonable ineffectiveness of the deeper layers. *arXiv preprint arXiv:2403.17887*, 2024.

Akshat Gupta, Jay Yeung, Gopala Anumanchipalli, and Anna Ivanova. How do llms use their depth? *arXiv preprint arXiv:2510.18871*, 2025.

Danny Halawi, Jean-Stanislas Denain, and Jacob Steinhardt. Overthinking the truth: Understanding how language models process false demonstrations. *arXiv preprint arXiv:2307.09476*, 2023.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Yi Hu, Cai Zhou, and Muhan Zhang. What affects the effective depth of large language models? *arXiv preprint arXiv:2512.14064*, 2025.

Vedang Lad, Jin Hwa Lee, Wes Gurnee, and Max Tegmark. The remarkable robustness of llms: Stages of inference? *arXiv preprint arXiv:2406.19384*, 2024.

Michael A Lepori, Michael Curtis Mozer, and Asma Ghandeharioun. Racing thoughts: Explaining contextualization errors in large language models. 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)*, pp. 3020–3036, 2025.

Ang Lv, Yuhan Chen, Kaiyi Zhang, Yulong Wang, Lifeng Liu, Ji-Rong Wen, Jian Xie, and Rui Yan. Interpreting key mechanisms of factual recall in transformer-based language models. *arXiv preprint arXiv:2403.19521*, 2024.

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

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. *arXiv preprint arXiv:2310.07923*, 2023.

William Merrill and Ashish Sabharwal. A little depth goes a long way: The expressive power of log-depth transformers. *arXiv preprint arXiv:2503.03961*, 2025.

William Merrill, Ashish Sabharwal, and Noah A Smith. Saturated transformers are constant-depth threshold circuits. *Transactions of the Association for Computational Linguistics*, 10:843–856, 2022.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. Language models implement simple word2vec-style vector arithmetic. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5030–5047, 2024.

nostalgebraist. Interpreting GPT: the logit lens. `https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens`, 2020. Accessed: 2026-02-18.

Enrique Queipo-de Llano, Álvaro Arroyo, Federico Barbero, Xiaowen Dong, Michael Bronstein, Yann LeCun, and Ravid Shwartz-Ziv. Attention sinks and compression valleys in llms are two sides of the same coin. *arXiv preprint arXiv:2510.06477*, 2025.

Clayton Sanford, Bahare Fatemi, Ethan Hall, Anton Tsitsulin, Mehran Kazemi, Jonathan Halcrow, Bryan Perozzi, and Vahab Mirrokni. Understanding transformer reasoning capabilities via graph algorithms. *Advances in Neural Information Processing Systems*, 37:78320–78370, 2024.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J Reddi. Reasoning with latent thoughts: On the power of looped transformers. *arXiv preprint arXiv:2502.17416*, 2025.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L Hamilton. Clutrr: A diagnostic benchmark for inductive reasoning from text. *arXiv preprint arXiv:1908.06177*, 2019.

Qi Sun, Marc Pickett, Aakash Kumar Nain, and Llion Jones. Transformer layers as painters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25219–25227, 2025.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, 2020.

Yiwei Wu, Atticus Geiger, and Raphaël Millière. How do transformers learn variable binding in symbolic programs? *arXiv preprint arXiv:2505.20896*, 2025.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. *arXiv preprint arXiv:2309.16042*, 2023.

A IMPLEMENTATION DETAILS

All models are accessed through the huggingface transformers library (Wolf et al., 2020); see Table 1 for their links. The logit lens analyses were implemented by accessing the intermediate hidden states and LM-head modules using the functionalities present in the transformers library directly, while the causal patching analyses were run using the nnsight (Fiotto-Kaufman et al., 2024) library.

Table 1: All pretrained models used in this study, with HuggingFace identifiers, parameter counts, and number of transformer layers.

Family	HuggingFace identifier	Parameters	Layers
GPT-2	`gpt2`	117M	12
	`gpt2-medium`	345M	24
	`gpt2-large`	774M	36
	`gpt2-xl`	1.5B	48
Pythia	`pythia-160m-deduped`	160M	12
	`pythia-410m-deduped`	410M	24
	`pythia-1.4b-deduped`	1.4B	24
	`pythia-2.8b-deduped`	2.8B	32
Phi	`microsoft/phi-1`	1.3B	24
	`microsoft/phi-1.5`	1.3B	24
	`microsoft/phi-2`	2.7B	32
	`microsoft/phi-4`	14B	40
	`microsoft/phi-4-reasoning`	14B	40
Qwen2	`Qwen/Qwen2-0.5B`	494M	24
	`Qwen/Qwen2-1.5B`	1.5B	28
	`Qwen/Qwen2-Math-1.5B`	1.5B	28
	`Qwen/Qwen2-7B`	7.6B	28
	`Qwen/Qwen2-Math-7B`	7.6B	28
Qwen2.5	`Qwen/Qwen2.5-0.5B`	494M	24
	`Qwen/Qwen2.5-1.5B`	1.5B	28
	`Qwen/Qwen2.5-Math-1.5B`	1.5B	28
	`deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B`	1.5B	28
	`Qwen/Qwen2.5-3B`	3.1B	36
	`Qwen/Qwen2.5-7B`	7.6B	28
	`Qwen/Qwen2.5-Math-7B`	7.6B	28
	`deepseek-ai/DeepSeek-R1-Distill-Qwen-7B`	7.6B	28
LLaMA	`meta-llama/Llama-3.2-1B`	1.2B	16
	`meta-llama/Llama-3.2-3B`	3.2B	28
	`meta-llama/Llama-3.1-8B`	8.0B	32

Finetuning implementation details. All models are finetuned using the TRL SFTTrainer with custom completion-only supervision: the cross-entropy loss is computed on the answer token T only. Training data is generated using the CLUTRR generator, producing 1000 examples consisting of an equal mixture of 2–5-hop or 2-10hop stories for each of the two settings. To prevent memorisation of individual names, all individuals are renamed to Person1 through PersonK, with the numerical assignment order randomised across examples. Stories are presented in plain completion format (ending with *Therefore, PersonA is PersonB's"*) without a chat template. For LoRA finetuning, we use rank $r = 16$, $\alpha = 32$, and dropout 0.05, with no bias adaptation, targeting the attention modules (using the QKV projection matrices through the merged `c_attn` module for GPT-2 only; and the `query_key_value` and the `dense` (output projection) modules for Pythia[8]). Full finetuning updates all model parameters. In both regimes, we train for 20 epochs with a learning rate of 2×10^{-4}, a batch size of 2 with gradient accumulation over 4 steps (effective batch size 8), a warmup of 50 steps, and a maximum sequence length of 512 tokens.

[8]We observed that including the output projection lead to better performance for the pythia models

B ADDITIONAL RESULTS

B.1 ADDITIONAL LOGIT-LENS RESULTS FOR PRETRAINED MODELS

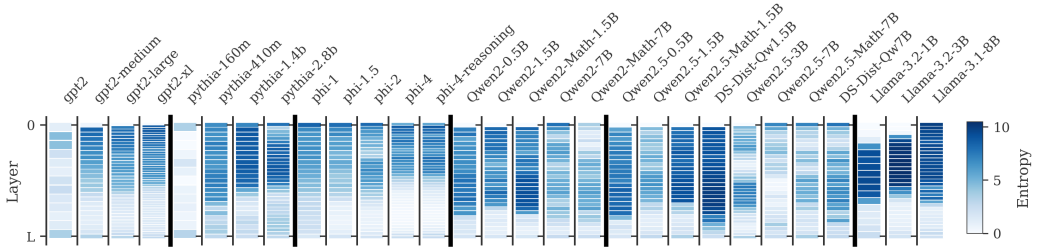
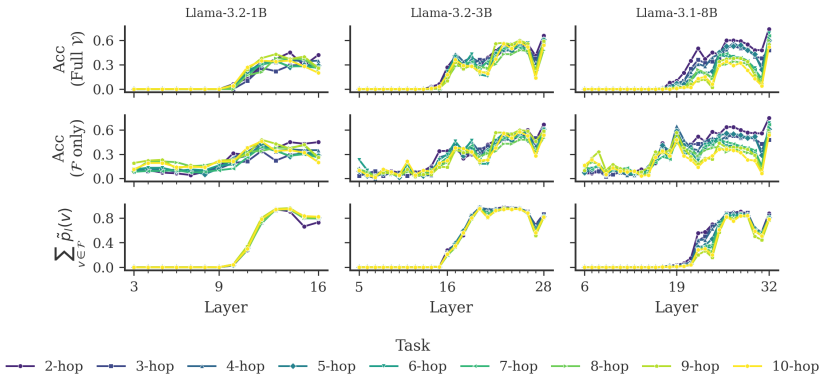
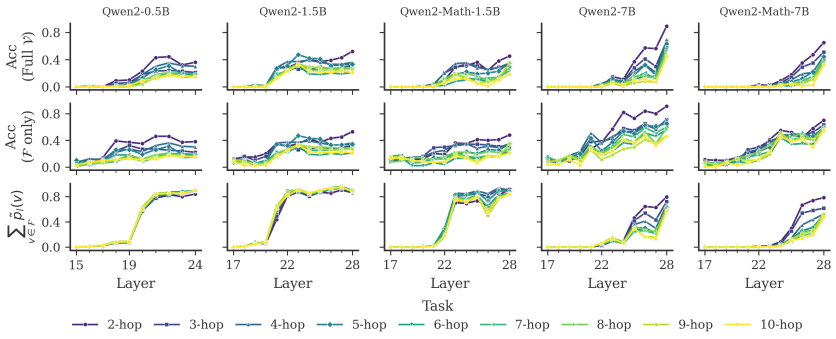


Figure B1: Entropy of the logit lens prediction $\tilde{p}_{l,T}$ by layer for pretrained models, averaged across all 900 CLUTRR 2-10hop test stories as in Fig. 2.



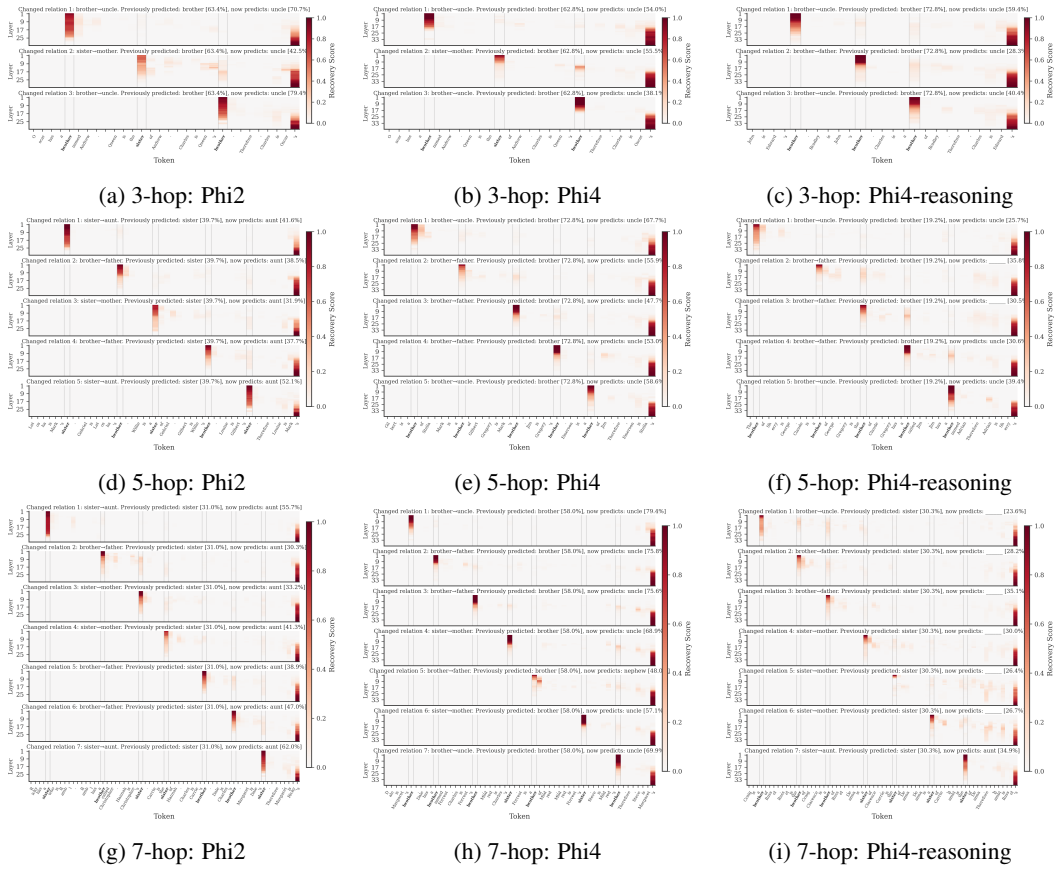
(a) Llama3 Family



(b) Qwen2 Family

Figure B2: Layerwise decoding results for Llama and Qwen2 families

B.2 Additional causal patching results for pretrained models (siblings-only setting)

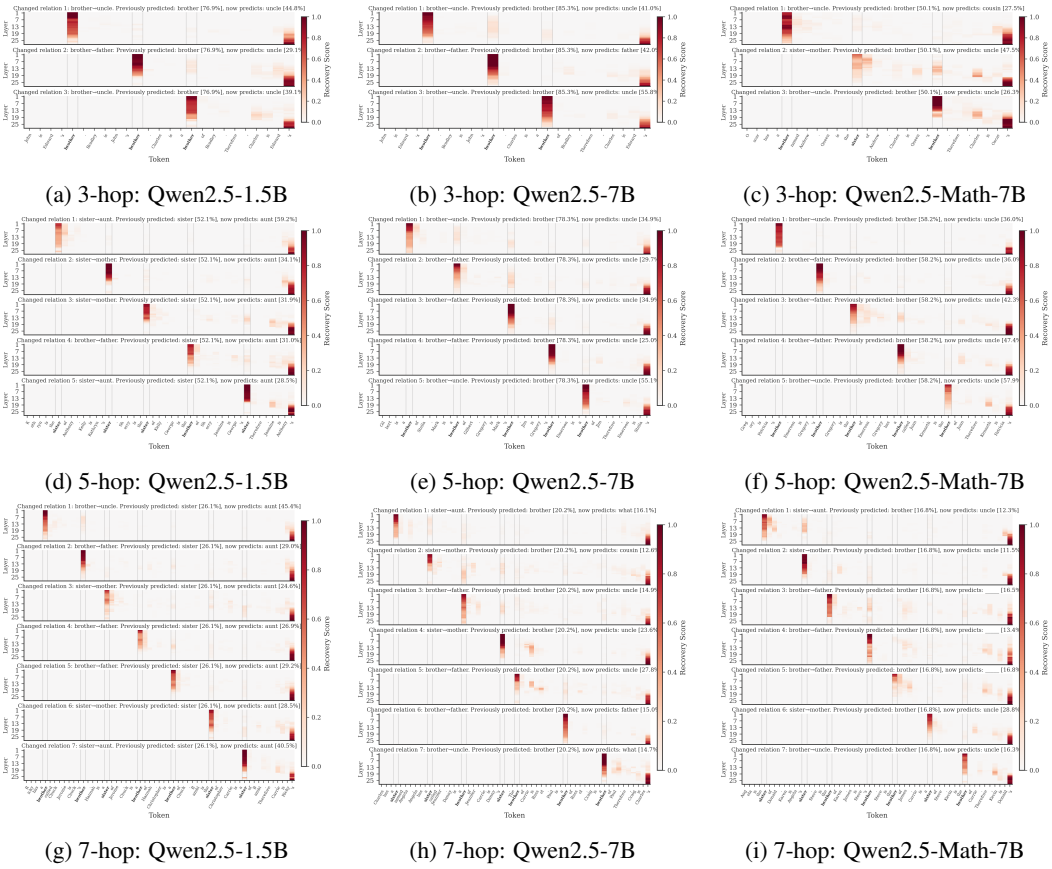


(a) 3-hop: Phi2 (b) 3-hop: Phi4 (c) 3-hop: Phi4-reasoning

(d) 5-hop: Phi2 (e) 5-hop: Phi4 (f) 5-hop: Phi4-reasoning

(g) 7-hop: Phi2 (h) 7-hop: Phi4 (i) 7-hop: Phi4-reasoning

Figure B3: Examples of causal patching trajectories for Phi models



(a) 3-hop: Qwen2.5-1.5B (b) 3-hop: Qwen2.5-7B (c) 3-hop: Qwen2.5-Math-7B

(d) 5-hop: Qwen2.5-1.5B (e) 5-hop: Qwen2.5-7B (f) 5-hop: Qwen2.5-Math-7B

(g) 7-hop: Qwen2.5-1.5B (h) 7-hop: Qwen2.5-7B (i) 7-hop: Qwen2.5-Math-7B

Figure B4: Examples of causal patching trajectories for Qwen2.5 models

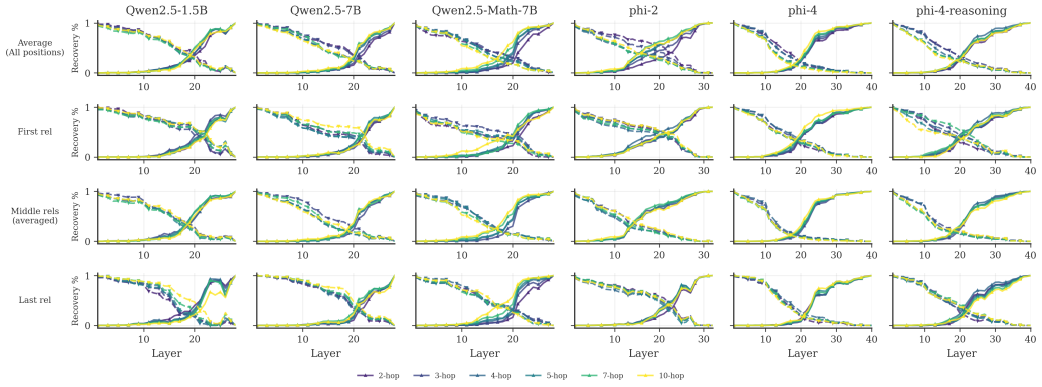


Figure B5: Average recovery score by relation replacement position, colored by number of hops, for stories which originally only had siblings. The first row is identical to Fig. 4(b) in the main text; this figure supplements the main text figure by splitting the averaged results across replacement positions.

B.3 Additional logit lens results for finetuned models

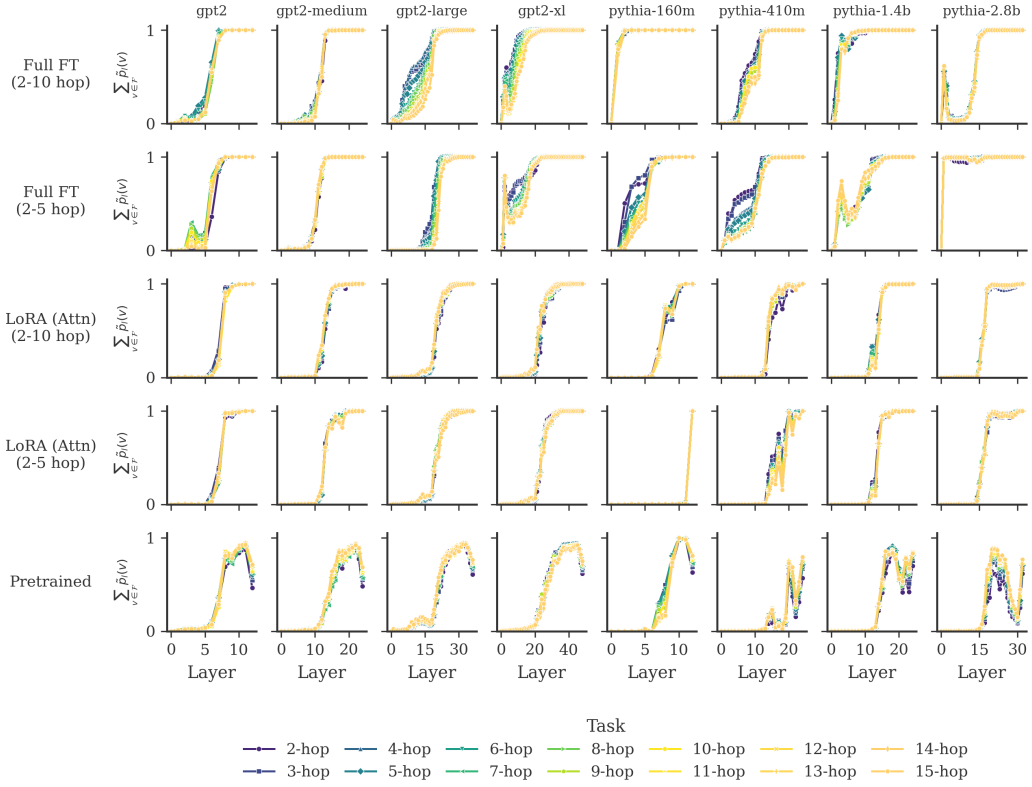
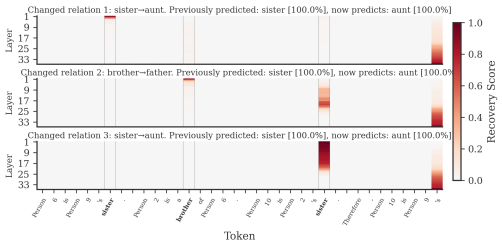
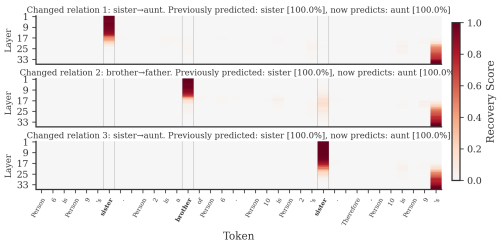


Figure B6: Probability assigned to family tokens by the logit lens prediction $\tilde{p}_{l,T}$ across layers and models by number of hops.

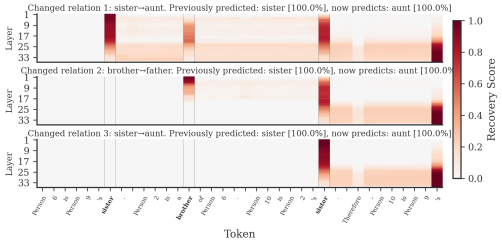
B.4 Additional causal patching results for finetuned models (siblings-only setting)



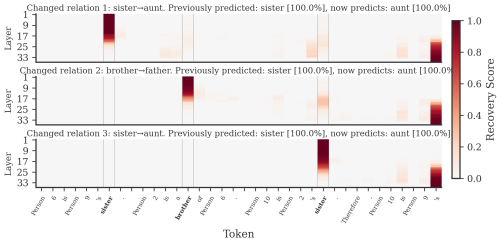
(a) Patch: $uncle \rightarrow brother$, full finetuning



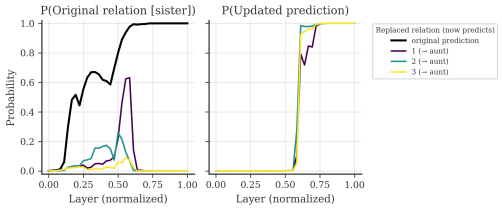
(b) Patch: $uncle \rightarrow brother$, LoRA finetuning



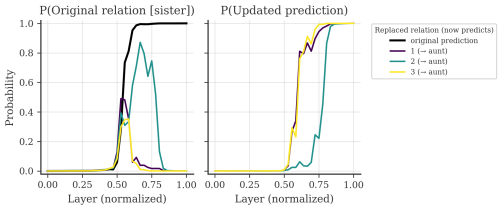
(c) Patch $brother \rightarrow uncle$, full finetuning



(d) Patch $brother \rightarrow uncle$, LoRA finetuning

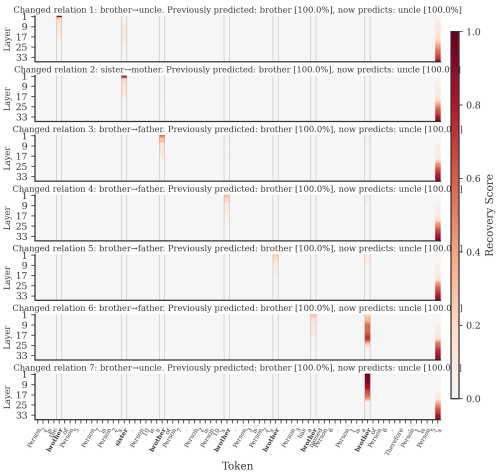


(e) Logit Lens: Full finetuning

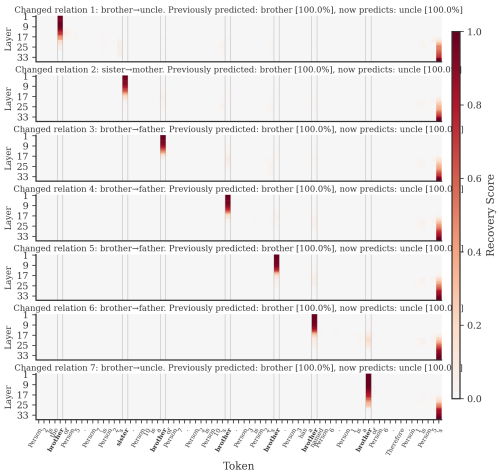


(f) Logit lens: LoRA finetuning

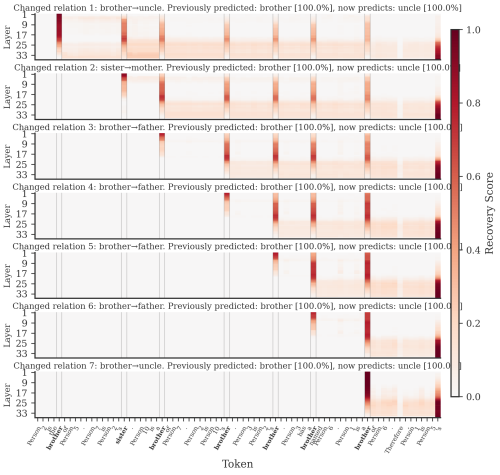
Figure B7: Full causal patching and logit-lens results for a 3-hop example using GPT2-large.



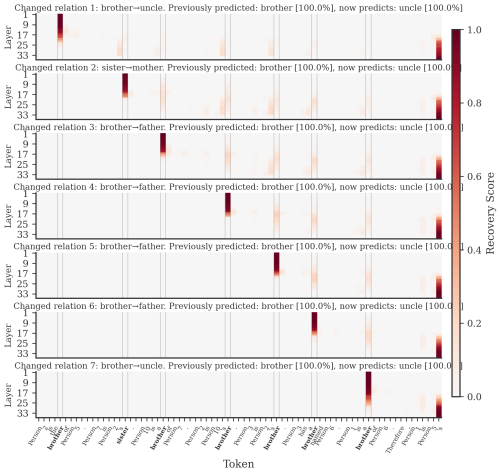
(a) Patch: $uncle \rightarrow brother$, full finetuning



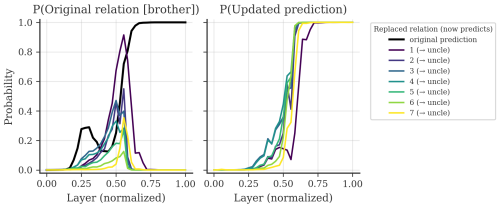
(b) Patch: $uncle \rightarrow brother$, LoRA finetuning



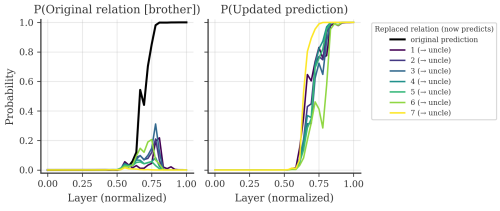
(c) Patch $brother \rightarrow uncle$, full finetuning



(d) Patch $brother \rightarrow uncle$, LoRA finetuning

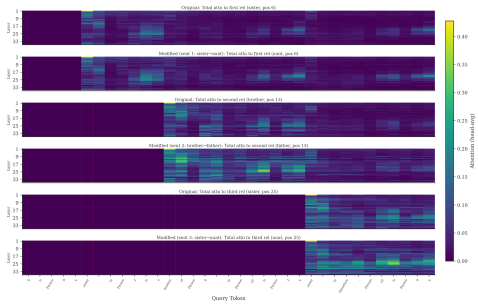


(e) Logit Lens: Full finetuning

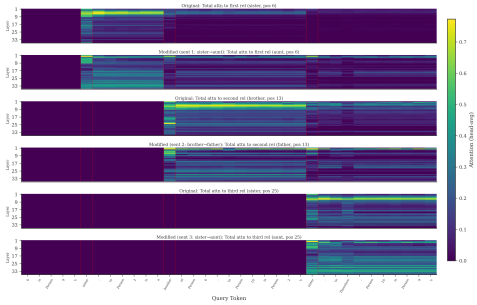


(f) Logit lens: LoRA finetuning

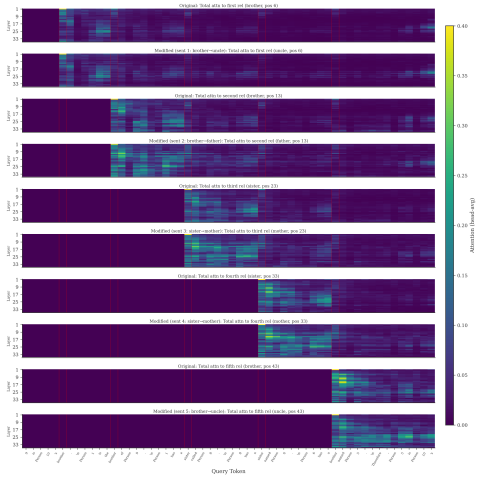
Figure B8: Full causal patching and logit-lens results for a 7-hop example using GPT2-large.



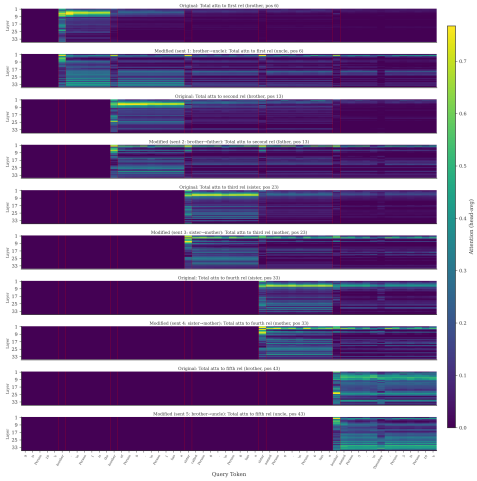
(a) 3-hop: LoRA finetuned



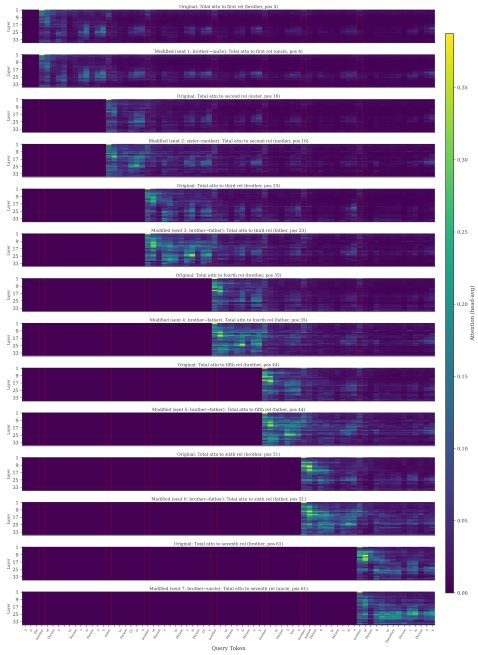
(b) 3-hop: Fully finetuned



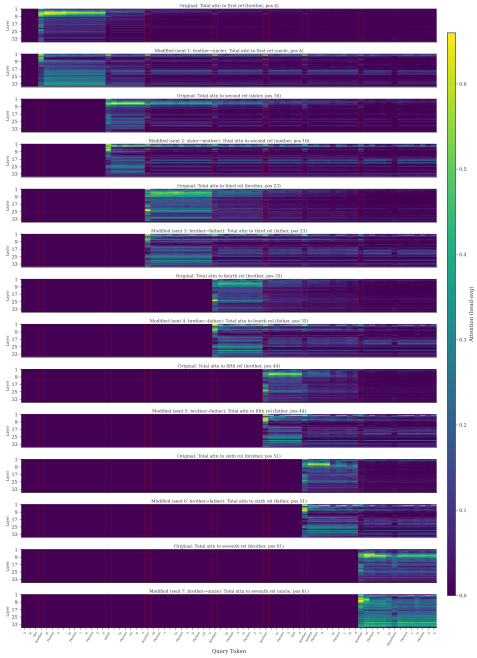
(c) 5-hop: LoRA finetuned



(d) 5-hop: Fully finetuned



(e) 7-hop: LoRA finetuned



(f) 7-hop: Fully finetuned

Figure B9: Attention to key token t^r by layer and query token for the causal patching examples of GPT2-large. Attention patterns differ between sibling and mother/father/uncle/aunt relations (first and second row of each location), more markedly so for the fully finetuned models.

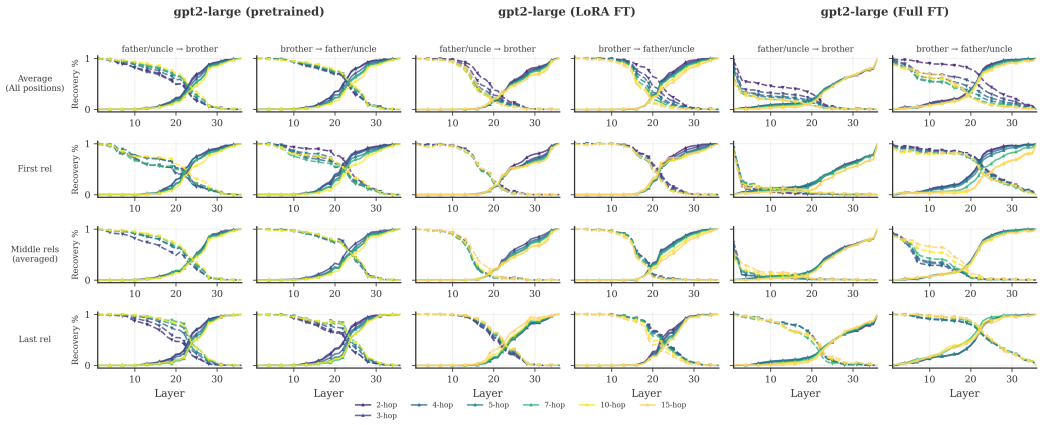
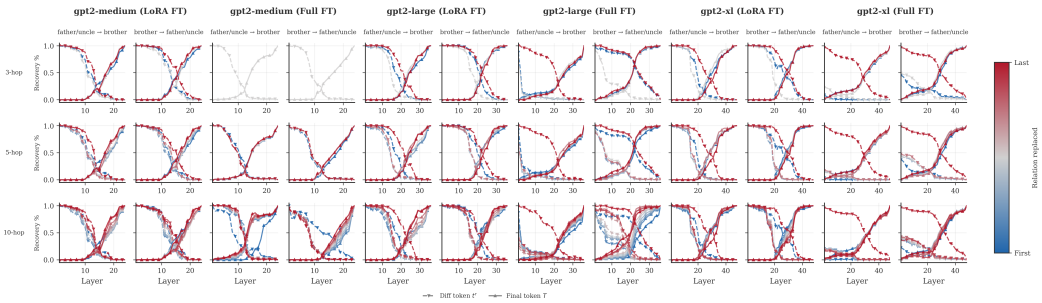
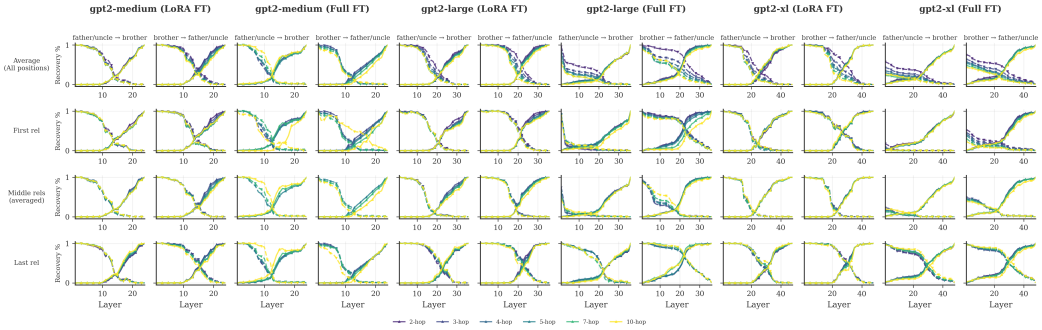


Figure B10: Average recovery score by relation replacement position for different versions of gpt2-large, colored by number of hops, for stories which originally only had siblings. The first row is identical to Fig. 7(b) in the main text; this figure supplements the main text figure by splitting the averaged results across replacement positions.

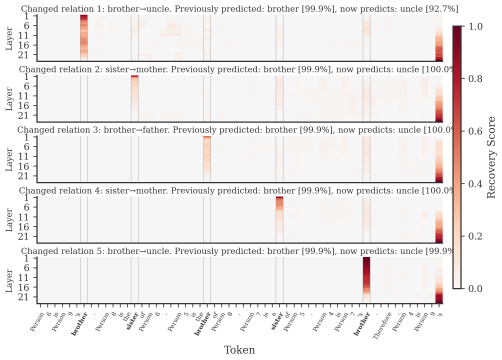


(a) Average recovery score by replaced relation across different hops (rows), colored by relation replaced.

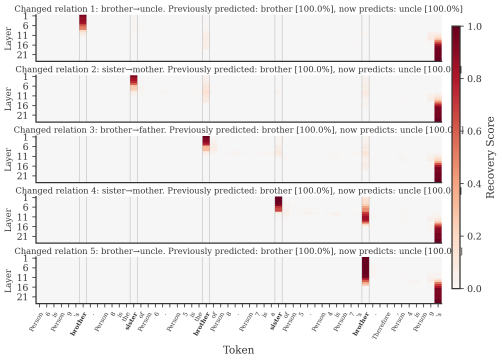


(b) Average recovery score across all relation replacement positions, colored by number of hops.

Figure B11: Causal patching analyses for finetuned GPT2 models of different sizes. Average recovery score at the replaced token t^r (dashed line) and the final token T (solid line) by model depth, for stories with only sibling relations. Note: For the finetuned GPT2-medium, the top prediction never flipped for some replaced relationships, leading to missing lines in both plots.

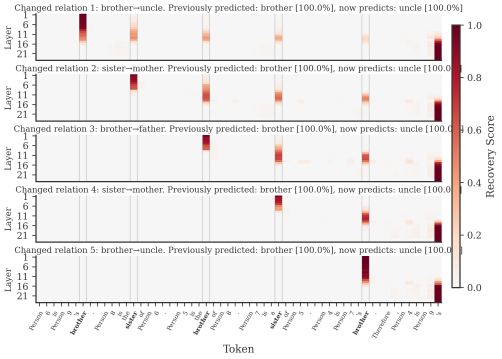


(a) Patch: $uncle \rightarrow brother$, full finetuning

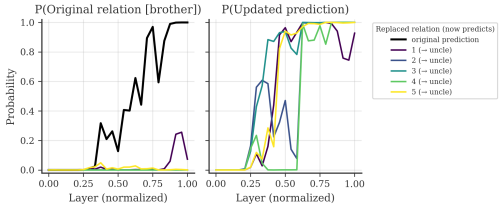


(b) Patch: $uncle \rightarrow brother$, LoRA finetuning

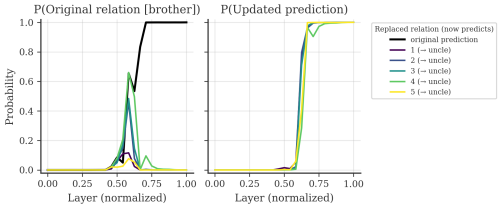
(c) Patch $brother \rightarrow uncle$, full finetuning



(d) Patch $brother \rightarrow uncle$, LoRA finetuning

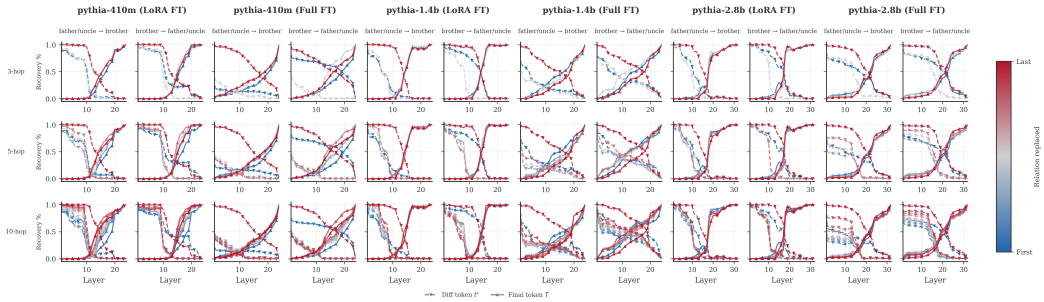


(e) Logit Lens: Full finetuning

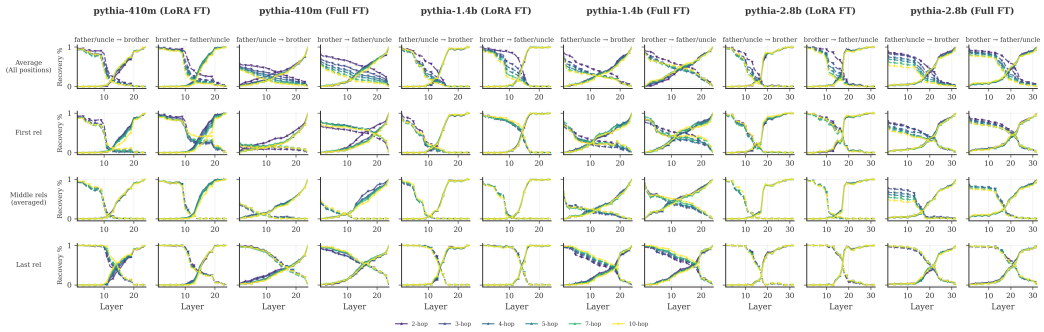


(f) Logit lens: LoRA finetuning

Figure B12: Full causal patching and logit-lens results for a 5-hop example using pythia-1.4B.



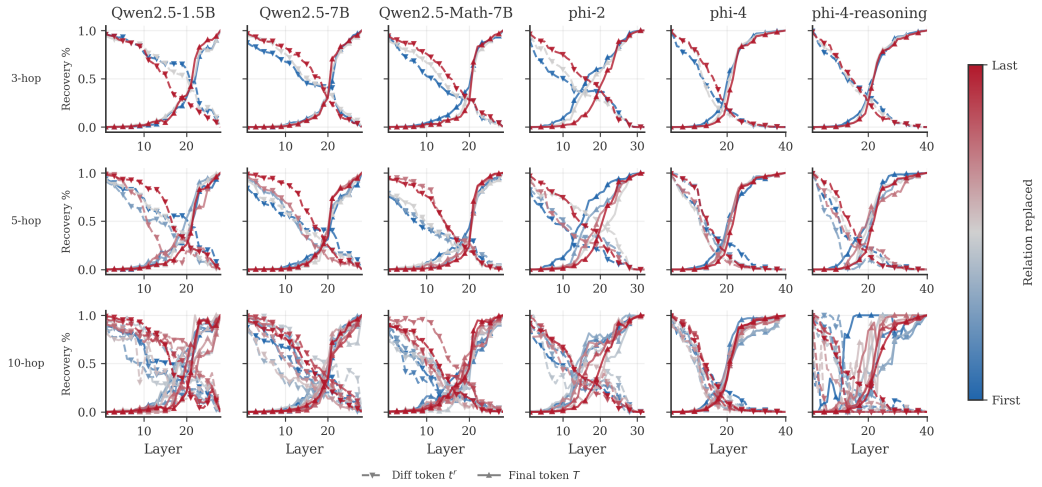
(a) Average recovery score by replaced relation across different hops (rows), colored by relation replaced.



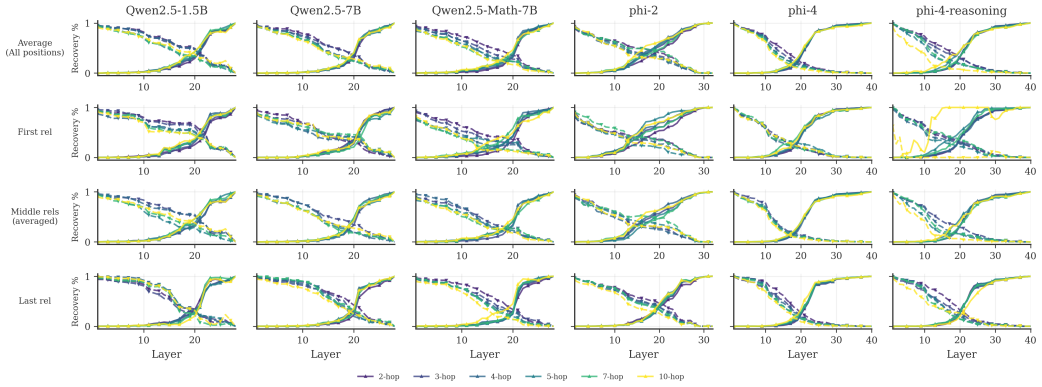
(b) Average recovery score across all relation replacement positions, colored by number of hops.

Figure B13: Causal patching analyses for finetuned pythia models. Average recovery score at the replaced token t^r (dashed line) and the final token T (solid line) by model depth, for stories with only sibling relations.

B.5 Additional causal patching results with more complex family relation stories

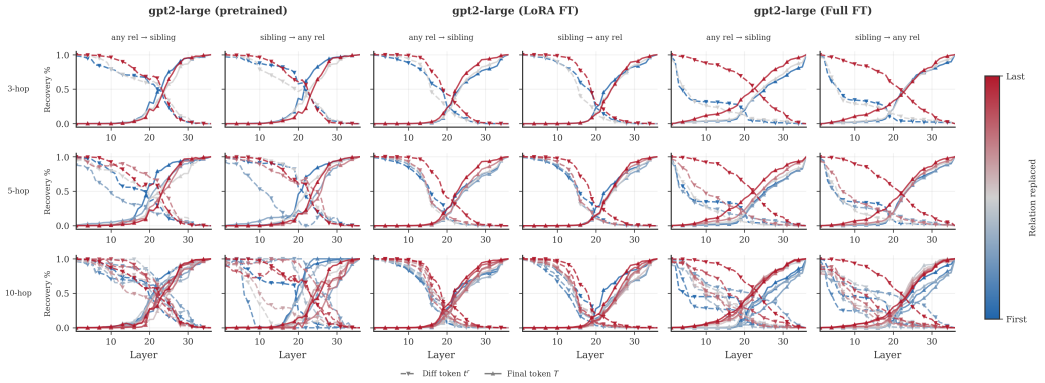


(a) Average recovery score by replaced relation across different hops (rows), colored by relation replaced.

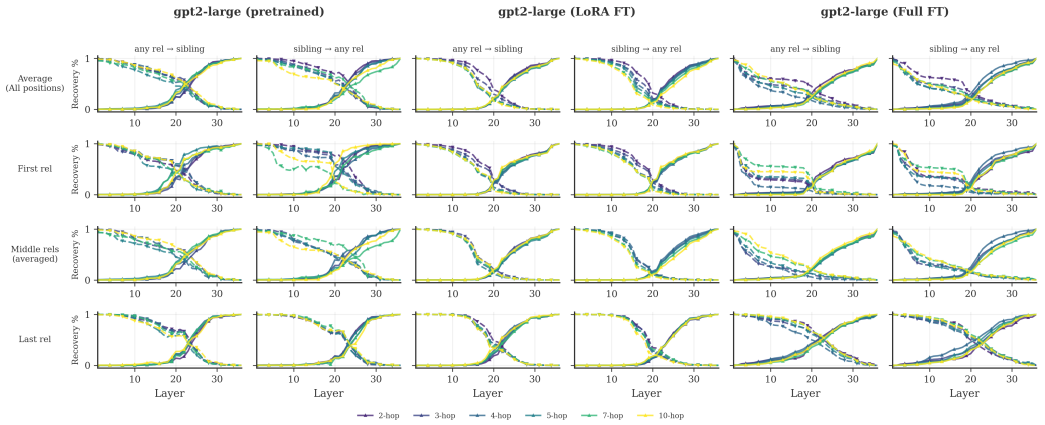


(b) Average recovery score across all relation replacement positions, colored by number of hops.

Figure B14: Causal patching analyses for pretrained models. Average recovery score at the replaced token t^r (dashed line) and the final token T (solid line) by model depth, for stories with all standard CLUTRR relation types, where existing relations are now replaced with "brother/sister". That is, this figure replicates the analyses in Fig. 4 without relying on stories that contain brother/sister relations only – with consistent trends.



(a) Average recovery score by replaced relation across different hops (rows), colored by relation replaced.



(b) Average recovery score across all relation replacement positions, colored by number of hops.

Figure B15: Causal patching analyses for GPT2-large. Average recovery score at the replaced token t^r (dashed line) and the final token T (solid line) by model depth, for stories with all standard CLUTRR relation types, where existing relations are now replaced with "brother/sister". That is, this figure replicates the analyses in Fig. 7 without relying on stories that contain brother/sister relations only. General trends are consistent with the main text except that now there is no visible difference between directions of patching. This is expected, as no direction is necessarily easier due to more involved family chains being present either way.

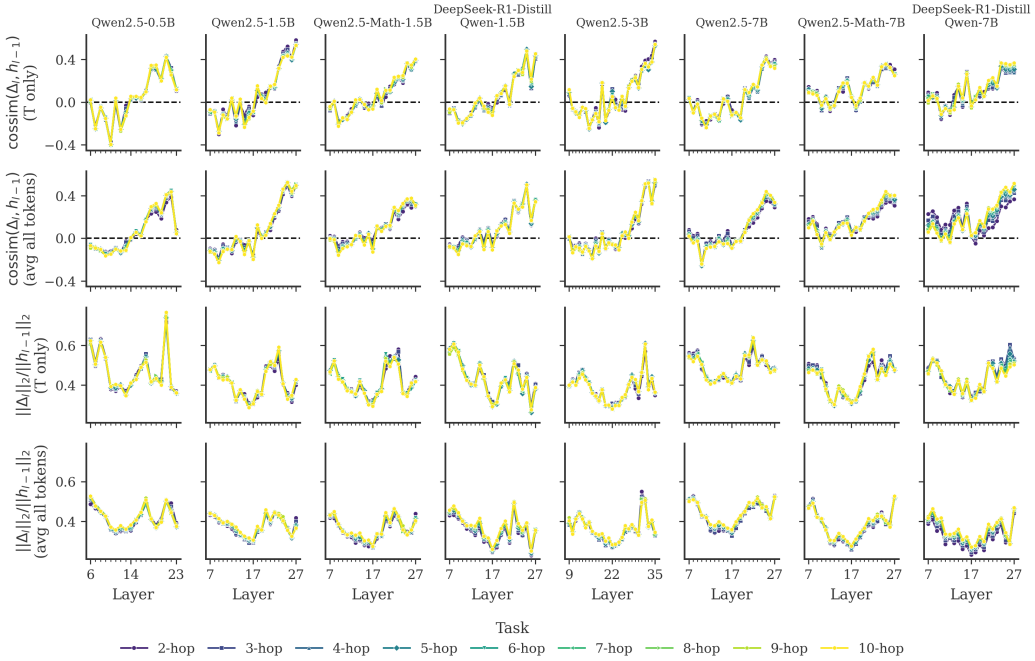
B.6 Additional analyses of hidden states

In this section, we analyse the behaviour of the hidden states directly, replicating part of the analyses of Csordás et al. (2025); Hu et al. (2025) in our setting. Specifically, we monitor the relative contribution of each layer $\Delta_l = h_l - h_{l-1}$ to the residual stream as $\frac{||\Delta_l||_2}{||h_{l-1}||_2}$, and the similarity of the update to the residual stream as $\text{cossim}(\Delta_l, h_{l-1})$. We compute these quantities both for the hidden state at the final token T only, and averaged across all tokens in the prompt. Note that across all plots we cut off the x-axis at the first 25% of layers and the final layer to ensure readability: as observed in Csordás et al. (2025), the contribution of the first few and final layers to the residual stream is much larger than that of intermediate layers, which makes smaller relative differences across hop counts indiscernible otherwise.

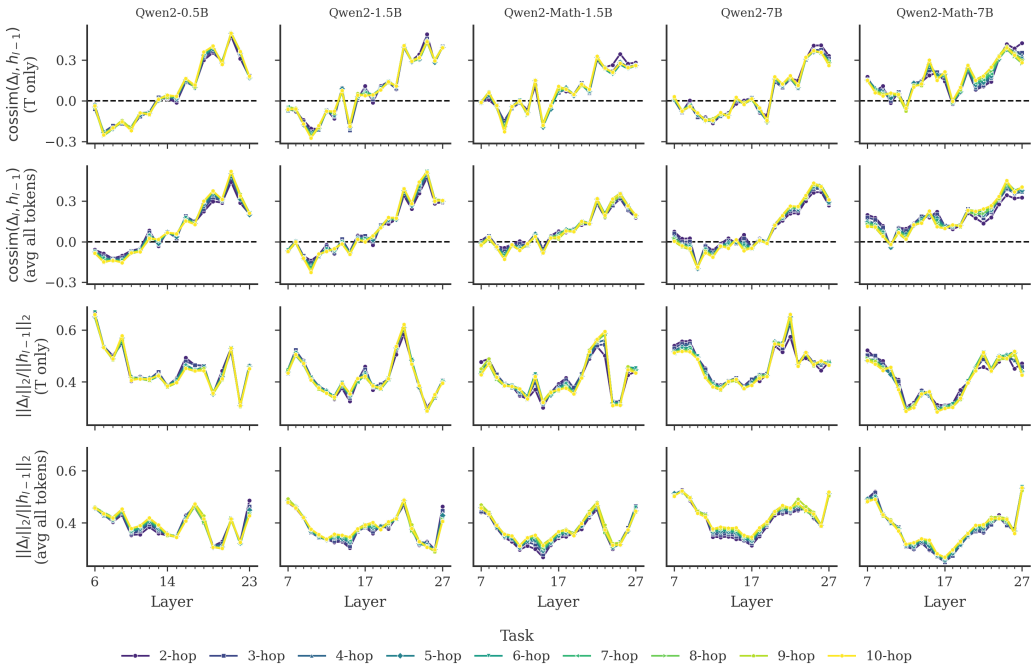
Across most pretrained models (Qwen2 and Qwen2.5 in Fig. B16, LLaMA and Phi in Fig. B17, GPT-2 in Fig. B18a, and Pythia in Fig. B19a), we observe the same general trends as Csordás et al. (2025): the contribution of later layers to the residual stream decreases in magnitude, and updates become increasingly similar in direction to the residual stream after being near-orthogonal in the middle layers. The main exception is Phi-1, where the cosine similarity trend appears reversed.

Regarding differences across task difficulty, the results are mixed and not strongly consistent across model families, although small differences are visible in all plots. In all families, the average contribution to the residual stream for higher-hop tasks appears slightly larger in some but not all of the middle to later layers when averaged across all tokens, though not at the final token alone (where one might expect most task-relevant computation to occur, given that the query is not known in advance). The novelty of computations, as measured by cosine similarity to the residual stream, tends to be higher (i.e. closer to zero) for lower hop counts when differences are present, with the exception of the pretrained GPT-2 and Pythia families, where cosine similarity is sometimes closer to zero for higher hop counts.

The results for the finetuned models, presented in Fig. B18 panels (b) and (c) for GPT-2 and Fig. B19 panels (b) and (c) for Pythia, are more clear-cut: across almost all model sizes and finetuning regimes, the cosine similarity between the residual stream and the layer update is markedly lower for higher hop counts, indicating that more novel computation is being performed for harder tasks.

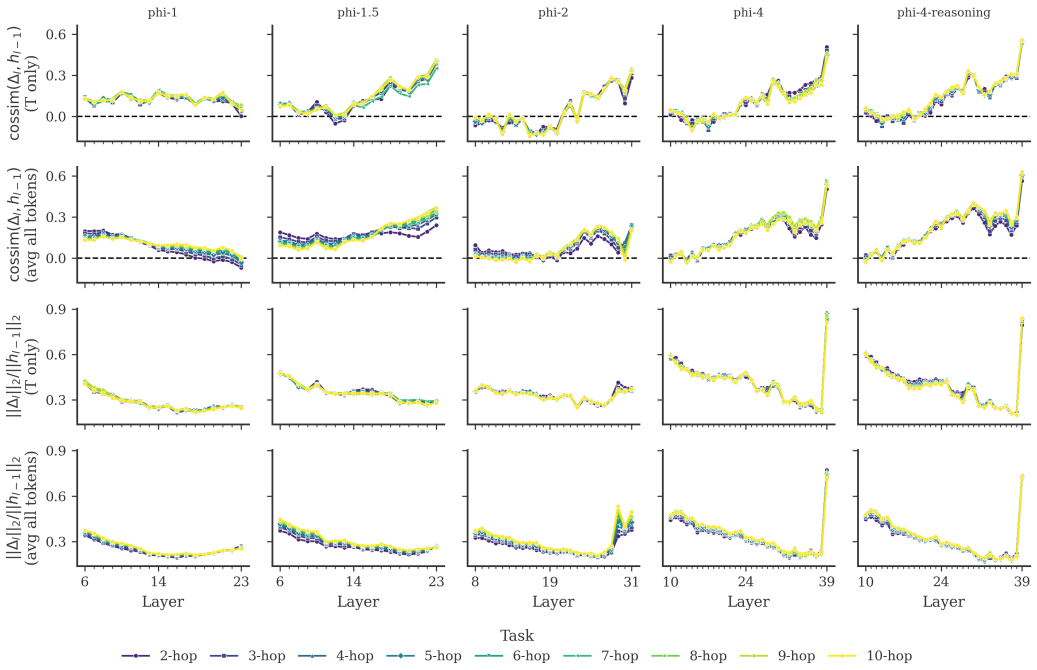


(a) Qwen2.5 family

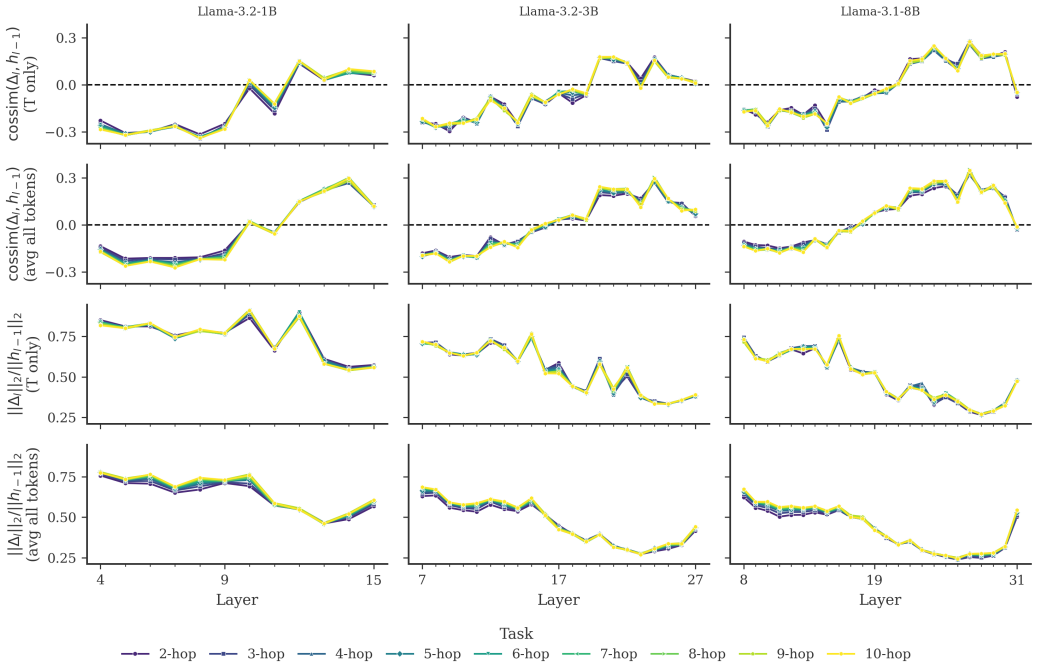


(b) Qwen2 family

Figure B16: Similarity of layer update to residual stream (top 2 rows) and relative contribution of layer update to residual stream (bottom 2 rows) by layer for Qwen models

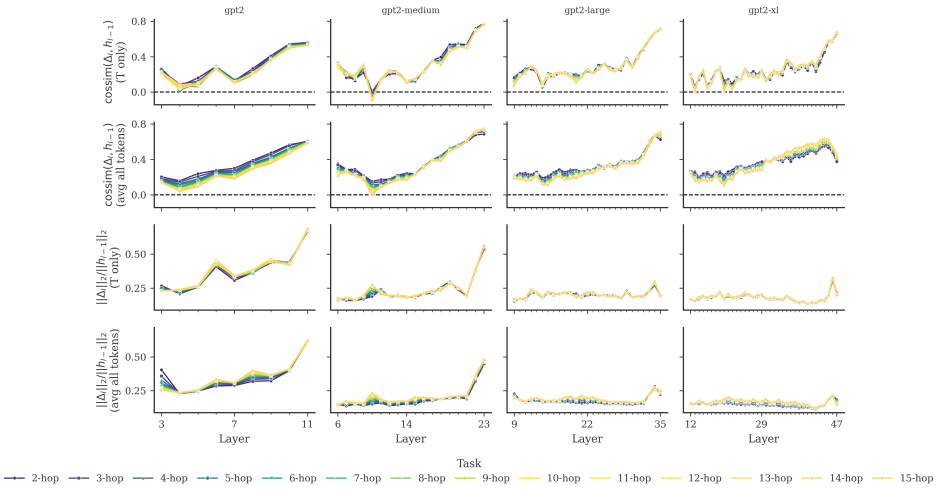


(a) Phi family

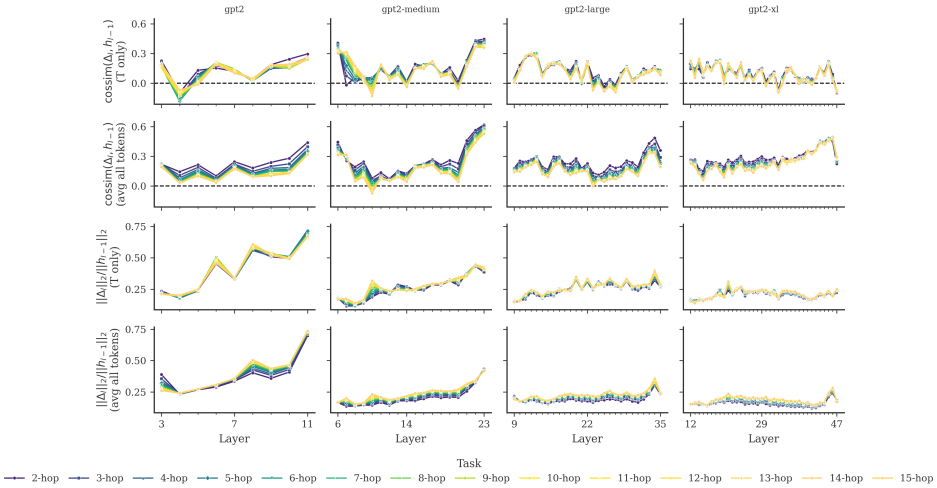


(b) Llama family

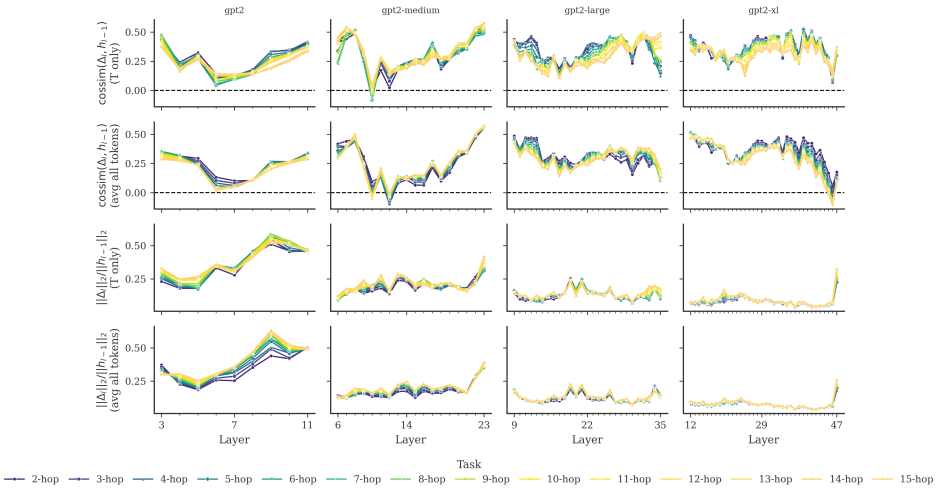
Figure B17: Similarity of layer update to residual stream (top 2 rows) and relative contribution of layer update to residual stream (bottom 2 rows) by layers for Phi and Llama models



(a) GPT2 Pretrained

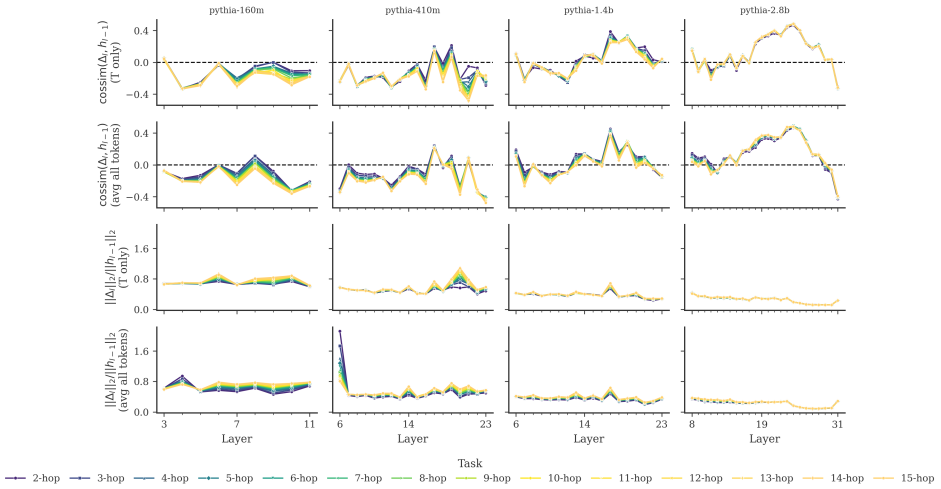


(b) GPT2 LoRA finetuned

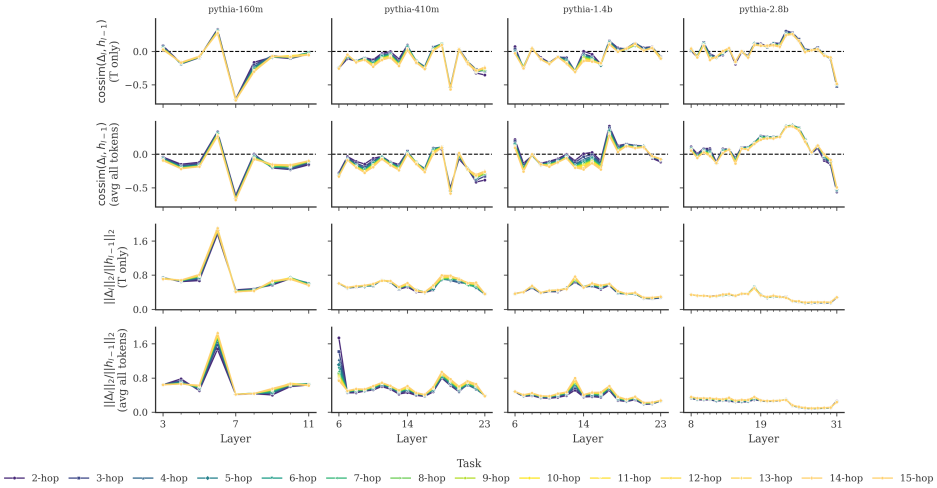


(c) GPT2 fully finetuned

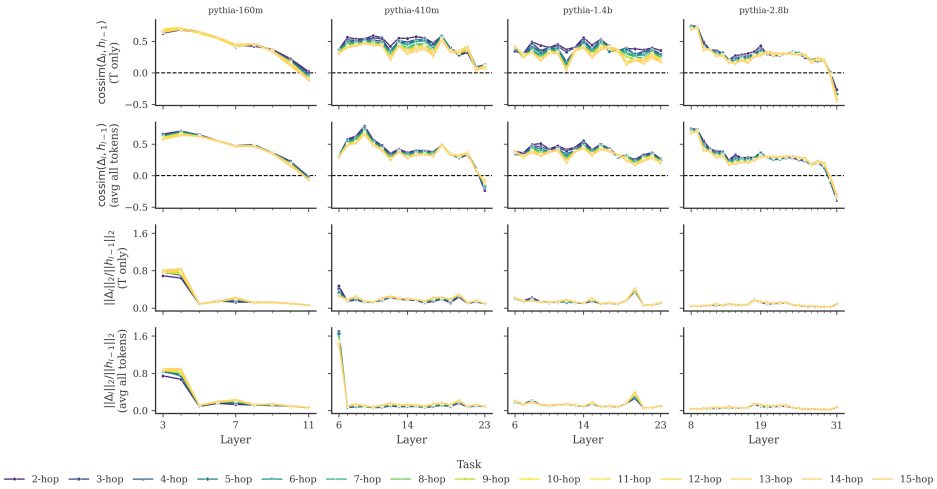
Figure B18: Similarity of layer update to residual stream (top 2 rows) and relative contribution of layer update to residual stream (bottom 2 rows) by layer for GPT2 family: Pretrained, LoRA finetuned and fully finetuned



(a) Pythia pretrained



(b) Pythia LoRA finetuned



(c) Pythia fully finetuned

Figure B19: Similarity of layer update to residual stream (top 2 rows) and relative contribution of layer update to residual stream (bottom 2 rows) by layer for pythia family: Pretrained, LoRA finetuned and fully finetuned

B.7 PERPLEXITY OF FINETUNED MODELS

Below, we evaluate the perplexity of finetuned models on the WikiText-2 dataset, confirming that the fully finetuned models do indeed lose their general language modeling ability entirely, while the LoRA finetuned models largely preserve it. We observe that for the smaller Pythia models, language modeling ability degrades under LoRA finetuning as well, whereas this does not occur for the smaller GPT-2 models. This is likely attributable to the difference in LoRA target modules: for Pythia, we include the attention output projection (`dense`) in addition to the QKV matrices, whereas for GPT-2 we target only the merged QKV projection (`c_attn`).

	Pretrained		LoRA (5-hop)		LoRA (10-hop)		FT (5-hop)		FT (10-hop)	
Model	CLUTRR (Ans.)	Wiki -2	CLUTRR (Ans.)	Wiki -2	CLUTRR (Ans.)	Wiki -2	CLUTRR (Ans.)	Wiki -2	CLUTRR (Ans.)	Wiki -2
GPT-2 (124M)	34.1	24.4	1.4	25.5	1.2	24.9	2.4	>1000	1.1	>1000
GPT-2 Medium (355M)	25.5	18.0	1.5	18.5	1.2	18.4	2.7	>1000	1.0	>1000
GPT-2 Large (774M)	20.4	15.6	1.1	15.7	1.1	15.7	1.9	>1000	1.0	>1000
GPT-2 XL (1.5B)	12.0	14.2	1.1	14.4	1.0	14.5	3.2	>1000	1.1	>1000
Pythia 160M	32.1	23.5	2.5	414.2	1.6	>1000	4.5	>1000	1.4	>1000
Pythia 410M	16.3	14.6	1.2	160.7	1.1	137.3	1.8	>1000	1.1	>1000
Pythia 1.4B	9.3	10.7	1.1	22.1	1.1	20.7	1.6	>1000	1.5	>1000
Pythia 2.8B	7.8	9.1	1.3	10.2	1.0	9.8	1.4	>1000	1.2	>1000

Table 2: Perplexity on CLUTRR final answer and WikiText-2 across model families and training conditions. CLUTRR = answer-only perplexity (averaged across hops 2–10); Wiki-2 = WikiText-2 test set perplexity (sliding window).