WHEN DO STORED FACTS BECOME AVAILABLE IN LOOPED TRANSFORMERS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Looped transformers do not acquire new knowledge during a forward pass, yet repeated applications of their shared block can change when and how stored knowledge becomes available. We study this process in looped transformers trained on synthetic tasks, focusing on factual retrieval and multi-hop reasoning. We first investigate single-hop factual retrieval. While prior work shows that looped and non-looped transformers store the same number of bits in their weights, it remains unclear how recurrence makes this knowledge available during inference. We find that stored facts become readable progressively across recurrences rather than through an immediate lookup. Moreover, the number of recurrences required to recover them increases with memorization load. Through representational analysis, we characterize this progression as a transition from representations of the query key to representations of its associated value. We then study multi-hop reasoning, where each retrieved entity serves as the key for the next relation. We find that intermediate entities emerge in logical order across recurrences, while the hidden-state geometry progressively shifts from the current frontier entity toward its successor. Comparing fixed- and dynamic-recurrence training, we find that dynamic recurrence exhibits faster state turnover and enables earlier solutions. We further show that these representations are functionally involved in the computation: counterfactual patches to the frontier entity redirect subsequent bridge entities and ultimately change the model's answer. Together, these results show that stored knowledge becomes progressively available in representations that support subsequent computation. More broadly, these findings suggest that recurrence organizes when and how stored knowledge can be used, rather than increasing storage capacity, motivating training and early-exit strategies that account for the progressive availability of knowledge across recurrences.

1 INTRODUCTION

Looped and non-looped transformers with same number of parameters have comparable memorization capacity (Zhu et al., 2025). This suggests that looping does not increase storage capacity of transformer-based architectures. However, existing analysis on memorization and knowledge manipulation typically examine the output at the last loop only (Zhu et al., 2025) overlooking a fundamental difference in how the two architectures process information. A looped model progressively refines the hidden representation through recurrence. Measuring only the final output therefore reveals little about what happens during the intermediate loops. This leaves open the central question addressed in this paper:

As a looped model runs, when does a stored fact become readable?

In a synthetic information-retrieval setting (Section 2), we find that memorized facts do not become readable immediately. Instead, knowledge emerges gradually across recurrent steps. Although the model's weights and stored content remain fixed, the amount of readable information increases with depth. Moreover, models trained to memorize more facts require additional recurrent steps to expose the same fraction of their total memorized information. To understand this delay, we analyze how the geometry of the hidden states evolves from the retrieval key toward the stored value. At each recurrent step, we measure whether representations cluster according to the key used in the query or

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

the corresponding memorized value. Early representations are organized primarily around the key, but gradually reorganize around the value as the recurrent computation proceeds. Under heavier memorization loads, this transition occurs later, suggesting that recurrence progressively constructs a value-aligned representation rather than simply reading out information that is already available.

We next study multi-hop reasoning (Section 3), where each retrieved entity serves as the key for the next relation in the chain. Intermediate entities become decodable in their logical order, following the same recurrence-dependent progression as final-answer accuracy. The key-to-value transition observed in single-hop retrieval repeats at each hop: representations initially encode the current frontier entity and then gradually shift toward its successor. We then compare a model trained with a fixed number of recurrences Kohli et al. (2026), with one trained using dynamically sampled recurrence counts Kohli et al. (2026). Although the two models achieve similar generalization performance, their internal mechanics differ. The dynamic model reaches the correct endpoint earlier and loses more quickly linear access to intermediate entities once they have been used. In contrast, the fixed depth one retains these entities across subsequent recurrent steps. This suggests that dynamic-recurrence training encourages functional state turnover, allowing the model to reuse its latent state as reasoning progresses. Finally, causal interventions show that these intermediate states are not merely correlated with the model's predictions, but actively contribute to the computation. Patching the representation of a frontier entity redirects the subsequent bridge entities and, ultimately, the final answer, while leaving the sequence of relations unchanged.

We summarize our contributions as follows:

- We provide a depth-resolved measure of readable memorised knowledge in bits, calibrated against a known source entropy (§2.1).
- We show that retrieval unfolds across recurrent steps rather than being a single lookup operation, and that the required depth increases with memorization load (§2.2).
- We characterize this retrieval process as a transition from representations of the query key to representations of the stored value (§2.3, §3.2).
- We show that intermediate entities in multihop chains become decodable in logical order, and that dynamic-recurrence training promotes earlier solvability and faster state turnover without compromising generalization (§3.2).
- Through causal interventions, we demonstrate that these intermediate representations guide subsequent bridge computations and the model's final answer (§3.3).

2 Retrieving stored facts

We begin with a controlled setting in which the model memorizes factual associations in its parameters. This setting allows us to study how stored knowledge becomes progressively readable across recurrent steps, how this retrieval schedule changes with increasing memorization load, and how the model's representations shift from the query key toward the corresponding stored value.

2.1 Experimental setup

Models, data, and measurement. Following the Ouro construction (Zhu et al., 2025), we train weight-tied, GPT-2-style looped Transformers with hidden size $d = 192$. The models' native readout consists of a final LayerNorm followed by the tied LM head. We consider loop budgets $L \in \{4, 8\}$ and train on $N \in \{20\text{K}, 50\text{K}, 100\text{K}\}$ synthetic samples.

Training applies the cross-entropy loss at the final loop only. An alternative would be to place a head after every loop and train all heads jointly. However, this would explicitly train the model to make the answer available at every depth. Finding the answer at intermediate depths would then reflect the training objective rather than reveal how retrieval naturally unfolds. By supervising only the final loop, we can study what the intermediate loops do without direct supervision.

To train our models, we use BioS (Allen-Zhu & Li, 2023), an established dataset of synthetic biographies in which each person's attributes are sampled uniformly from fixed sets of cities, universities, employers, majors, and birth dates (see a prompt-answer example in Appendix C). Because the size of each set is known, we can compute the uncertainty associated with uniform guessing. For an

attribute with K possible values, this uncertainty is $H_0 = \log_2 K$ bits. For example, choosing uniformly among 200 cities gives $H_0 \approx 7.64$ bits. After loop t, we query the model for an attribute and compute $\mathrm{CE}(t)$, the average negative log-probability assigned to the correct answer. We then define: $M(t) = H_0 - \mathrm{CE}(t)$.. This quantity measures how much the model improves over uniform guessing. At chance level, $\mathrm{CE}(t) = H_0$ and therefore $M(t) = 0$. If the model assigns full probability to the correct answer, then $\mathrm{CE}(t) = 0$ and $M(t) = H_0$. Values of $M(t) < 0$ occur when the model assigns a lower probability to the correct answer than a uniform guess would. Such values indicate that the model's prediction is biased away from the correct answer rather than merely uncertain.

Reading the model at intermediate depths. The native readout is trained only at the end of the final loop, so there is no reason to expect it to interpret intermediate hidden states correctly. We therefore read each depth using a *tuned lens* (Belrose et al., 2023), implemented as an affine transformation inserted before the model's native readout: $z^{(t)} = \texttt{lm_head}\big(\texttt{ln}_f\big(W_t h^{(t)} + b_t\big)\big)$, where $h^{(t)}$ denotes the hidden state after t loops, taken at the point where it would normally be passed to the native readout. The final normalization $\texttt{ln}_f$ and tied projection `lm_head` remain frozen. Only $W_t \in \mathbb{R}^{d\times d}$ and $b_t \in \mathbb{R}^d$ are learned, with a separate pair for each depth. Because the output projection is never retrained, the lens cannot learn a new mapping from representations to text. It can only transform an intermediate state into a basis that the model's existing readout can interpret. For comparison, we report results from both the trained tuned lens and the untrained native readout, which applies the model's frozen language head directly to each intermediate state. Each lens is trained and evaluated on disjoint data splits. For clarity, we use *tuned lens* to denote a classifier followed by the model's language head, and *linear probe* to denote a classifier applied directly to a representation space without the language head (which will be used in Section 3).

2.2 More stored facts require deeper retrieval

We first confirm the finding of Zhu et al. (2025) that increasing the number of loops does not increase memorization capacity. Across all parameter counts, the loop-1 (baseline that loops only once, i.e. a classical non-looped transformer), loop-4, and loop-8 models store the same total number of bits (Figure 5; Appendix A). Having established that recurrence does not increase storage capacity, we now examine how it shapes the process through which stored facts are retrieved.

Facts become readable progressively. The top row of Figure 1 shows the central phenomenon. The number of readable bits increases monotonically with depth and plateaus before the model reaches its trained loop count. By the final depth, all five attributes reach approximately 98% of their respective H_0 values in every setting. The model itself does not change along the x-axis: its weights and stored content remain fixed, and only the number of times the shared block has been applied varies. Retrieval is therefore not completed in the first loop, but unfolds gradually across multiple recurrent steps.

Depth scales with load. The bottom two panels of Figure 1 show how retrieval changes with memorization load. The left panel reports the readable bits for each attribute at the final depth as a function of dataset size. The curves remain flat, with the loop-4 and loop-8 models overlapping. A model trained on 100K individuals ultimately retrieves each fact as well as one trained on 20K individuals, while the loop-8 model reaches the same final performance as the loop-4 model. This recovers the storage result of Zhu et al. (2025) at the level of individual attributes. The right panel instead shows the depth required to reach that final level. Completion depth increases steadily with N for both loop budgets and across all attributes. Importantly, the two panels describe the same six models: their final retrieval performance remains unchanged, but the number of recurrent steps required to reach it increases with memorization load. Heavier loads therefore push retrieval toward later depths, while a larger loop budget spreads the same retrieval process across more recurrent steps without improving final accuracy.

Takeaway: *Looping does not increase how much the model memorizes, but it changes when that knowledge becomes readable.* For training. *More loops spread retrieval over greater depth rather than raising capacity if no intermediate loop supervision is used.* For early exit. *The required exit depth increases with memorization load. Facts can also become decodable before they appear in the native logits, so logit-based exit rules may detect them late.*

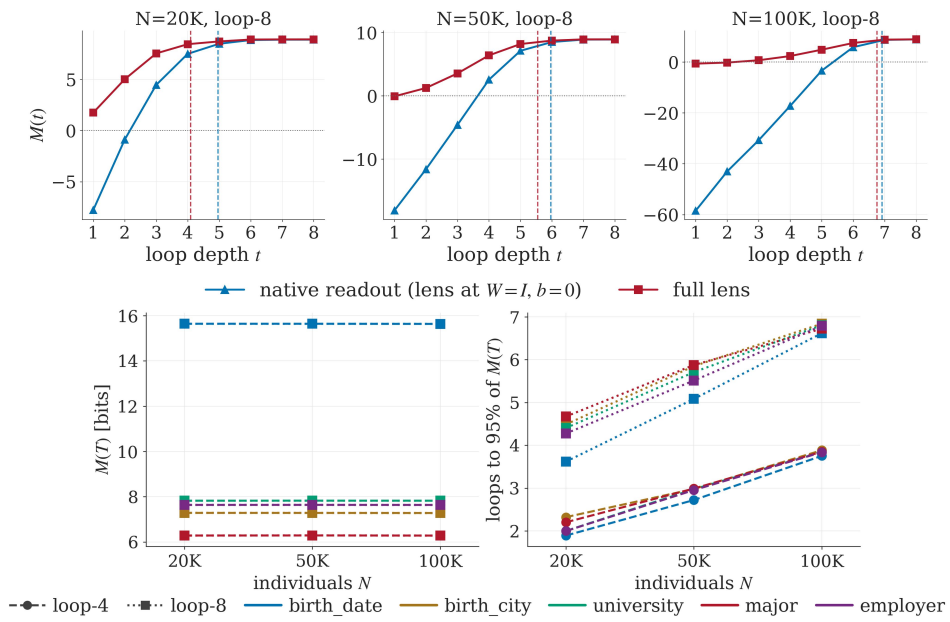


Figure 1: **Top:** $M(t) = H_0 - \mathrm{CE}(t)$ (nats) averaged over the five scored attributes for both, tuned lens (red curves) and the native readout (blue curves). Dashed vertical lines mark the plateau depth, the smallest t reaching 95% of that curve's own final value. **Bottom-Left:** readable bits at the final depth against dataset size. As expected, the lines of loop-4 and loop-8 coincide. **Bottom-Right:** depth required to recover 95% of $M(T)$. It rises with N at both loop budgets, for every attribute.

2.3 What the loop computes: from key to value

While $M(t)$ measures how many bits are readable after t recurrent steps, it does not reveal what the residual stream represents along the way. In particular, a state that has identified *which individual was named* and one that has recovered *the requested answer* are evaluated only through cross-entropy. Distinguishing between these two types of information reveals that the retrieval schedule observed in §2.2 reflects a gradual conversion from the former to the latter.

Measuring the key-to-value transition. To track this conversion, we measure whether the hidden states are organized according to the person named in the prompt, which serves as the retrieval key, or according to the corresponding attribute value. For each experimental condition, we sample $P = 1000$ individuals and generate $K = 6$ independent paraphrases for each individual. At every depth t, we extract the residual stream at the answer-token position and then center and scale each coordinate. We exclude birth dates because they contain multiple tokens. For a partition $\mathcal{G}$ of the resulting $n = PK$ states, we compute the trace form of the one-way MANOVA ratio:

$$F(\mathcal{G}; t) = \frac{\frac{1}{G-1}\sum_g n_g \|\bar{h}_g - \bar{h}\|_2^2}{\frac{1}{n-G}\sum_g \sum_{i \in g} \|h_i - \bar{h}_g\|_2^2}, \tag{1}$$

where G is the number of groups, n_g is the size of group g, h_i is sample i hidden state, $\bar{h}_g$ is the mean state of group g, and $\bar{h}$ is the global mean. The ratio is large when states within the same group cluster together and different groups remain well separated. We apply this measure to the same states using two different partitions. The *key score*, F_{key}, groups states by individual and measures how strongly they retain the identity provided in the prompt. The *value score*, F_{value}, groups states by the correct attribute value and measures whether states associated with the same answer converge toward a common region. Because the person's identity is included in the prompt, a high F_{key} does not require memorized knowledge. By contrast, the answer is not present in the prompt, so F_{value} measures retrieval from the model's weights. Because each individual contributes K correlated paraphrases, the null value of F_{value} is greater than one. We therefore normalize it using a permutation baseline that shuffles the mapping between individuals and values while preserving

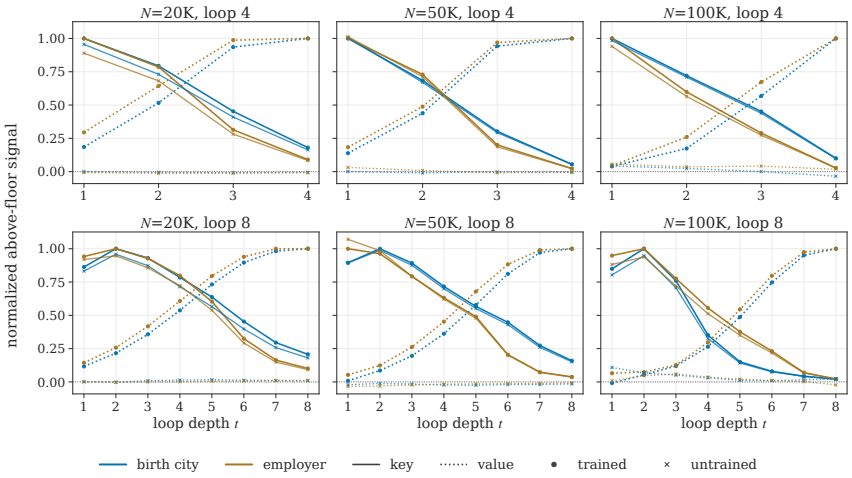


Figure 2: Normalized F_{key} and F_{value} across recurrent steps and memorization loads for the trained and untrained groups.

paraphrase blocks and group sizes. As a negative control, we repeat the analysis on a group whose facts were never seen by the model.

The loop turns a key into a value. Figure 2 reports the key and value scores computed from the same states. Early in the computation, the states are organized primarily around the individual named in the prompt. As the recurrent computation proceeds, this key-centered structure declines and the states become increasingly organized around the corresponding attribute value. The control group separates these two stages. Individuals whose facts were never memorized follow the same key trajectory, but their value score remains at the empirical floor. The key structure is therefore driven by information in the prompt, whereas the emerging value structure reflects retrieval from the model's weights. This separation is most apparent under a heavy memorization load. After the first recurrence, the individual is already strongly represented, while the associated value remains indistinguishable from the untrained floor. The model therefore resolves the retrieval key before the answer becomes available. Over subsequent recurrences, the representation gradually shifts from the key toward the stored value. We define the retrieval midpoint $t_{1/2}$ as the depth at which the value score has completed half of its total increase above the empirical floor. After normalizing by the loop budget L, the midpoint occurs near $0.52L$ for 20K individuals, around $0.60L$ for 50K individuals, and between $0.65L$ and $0.72L$ for 100K individuals. This pattern is consistent across the loop-4 and loop-8 models and across all single-token attributes. Heavier memorization loads therefore shift the key-to-value transition toward the end of the available recurrent depth.

Takeaway: *Across recurrent steps, the model's representation shifts from the retrieval key to the memorized value. Heavier memorization loads cause this transition to occur later.*

3 Composing Stored Facts

A central motivation for recurrent depth is that successive recurrences may carry out successive steps of an implicit computation. This raises a natural question: *how does factual information emerge over the course of multi-hop reasoning in looped transformers?*. To address this question, we track intermediate entities along a multi-hop chain, examining when they become represented and how each entity transitions from the value retrieved at one hop to the key used at the next. Finally, we use causal interventions to test whether these intermediate states influence the model's subsequent computation and final prediction.

3.1 Experimental setup

We use pre-trained models from Kohli et al. (2026), trained with either fixed, R6 (6 loops), or dynamic, R_dyn, numbers of recurrent loops. These models share a common architecture with

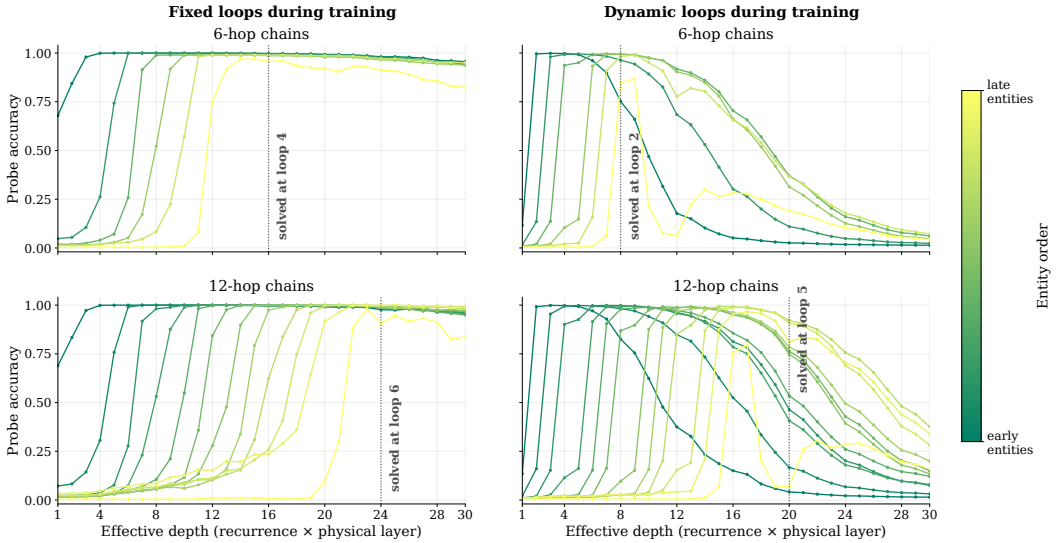


Figure 3: **Dynamic recurrence accelerates multihop reasoning.** Probe accuracy for 6-hop (top) and 12-hop (bottom) chains in R6 (left) and R_dyn (right). Colors indicate entity order and dotted lines mark the loop at which endpoint accuracy reaches 80%.

a four-layer recurrent block and generalize to unseen loop counts, providing a natural setting for studying how multi-hop computations unfold with recurrent depth. A k-hop query is presented as a flat sequence: $(e_0, r_1, \ldots, r_k)$, where each entity e_i and each relation r_j is a token. Each sequence defines a unique chain $e_0 \xrightarrow{r_1} e_1 \xrightarrow{r_2} \cdots \xrightarrow{r_k} e_k$. The model is trained to emit only the endpoint e_k and the bridge entities $e_1, \ldots, e_{k-1}$ are never supervised as outputs for that composite query (see a prompt-answer example in Appendix C).

Intermediate-entity probes. To determine when each intermediate entity becomes decodable, we train linear probes on the model's hidden states. Unlike in the single-hop setting in Section 2, neither the language-model head nor a linear transformation followed by a linear head provide an effective readout. Our experiments show that even a linear classifier applied to the final input-token state cannot recover all the relevant information. We therefore apply our primary probes to the concatenated hidden states from all token positions in the chain. We train a separate probe for each recurrent loop, physical layer, and hop index, treating each loop–layer pair as an effective depth. Each probe predicts the entity token through a vector of vocabulary logits. We run the R6 checkpoint for 10 loops and extract activations after every physical layer. The probe dataset contains approximately 50,000 samples drawn from the Kohli et al. (2026) test split. We use chains containing up to 25 hops, but construct probe datapoints only for intermediate entities at hop indices 2–15, always excluding the chain's final hop. Because R_dyn pads every input to 50 tokens, we mask positions outside the actual chain before concatenating the states. Probes are trained for 10 epochs. As a complementary local readout, we also train per-token probes for every loop, layer, and hop, using each token's hidden state separately instead of concatenating states across the sequence.

3.2 Information retrieval in multi-hop chains

Looped models exhibit latent reasoning. We use the trained probes to track when each intermediate entity becomes decodable across the model's unrolled recurrent depth. Figure 3 shows these decoding trajectories for chains of lengths 6 and 12 for both R6 and R_dyn. The intermediate entities emerge in a consistent order that mirrors the logical structure of the underlying chain. This provides direct evidence of latent sequential reasoning: although bridge entities are never supervised as outputs of the multi-hop task, they become linearly decodable in the order in which they are required to solve it. Rather than exposing all intermediate information simultaneously, the model progressively reveals successive hops as recurrent depth increases.

Dynamic recurrence supports faster state turnover. The two training regimes differ in how long these intermediate representations remain accessible. In R6, earlier entities remain linearly recoverable after later entities become decodable, suggesting that multiple intermediate variables coexist in the hidden state. By contrast, R_dyn progressively loses linear access to earlier entities after they have contributed to subsequent hops, indicating more aggressive reuse of its representational capacity. These different internal dynamics arise even though both models exhibit similar generalization capabilities. This faster turnover also coincides with earlier endpoint solvability. For the 6-hop and 12-hop tasks, R_dyn reaches the accuracy threshold at loops 2 and 5, respectively, whereas R6 reaches it at loops 4 and 6. Forgetting an intermediate entity therefore does not necessarily indicate a failure of reasoning. Under dynamic-loop training, it may instead reflect a strategy in which intermediate information is retained only for as long as it remains computationally useful.

Takeaway: *These results suggest that dynamic-loop training provides a useful inductive bias for multihop reasoning, preserving generalization while promoting earlier solutions and more efficient reuse of latent state.*

From key to value in reasoning chains.

The key-to-value transition observed in Section 2.3 also recurs during multi-hop composition. To characterize the transition from e_j to e_{j+1}, at each layer we concatenate the hidden states at the token positions corresponding to r_j and r_{j+1}. We align these representations by the layer at which e_j first becomes decodable and use the permutation-normalized MANOVA metric from Section 2.3 to track the signals for e_j and e_{j+1}.

As shown in the right plot of Figure 6, the signal for the current entity initially dominates but subsequently declines, while the signal for the next entity grows and eventually overtakes it. Thus, once e_j becomes available, the following layers appear to use it as a key—together with the next relation—to retrieve e_{j+1} as the corresponding value. Multi-hop reasoning therefore repeatedly instantiates a similar key-to-value transformation, advancing the computation from one bridge entity to the next.

Takeaway: *Looped transformers are capable of latent multi-step reasoning, iteratively executing successive entity-retrieval operations across recurrent depth. Each step implements repeated key-to-value transitions, while the persistence of earlier intermediate entities depends on the recurrence-training regime.* For training. *Varying the recurrence depth during training appears to provide a promising inductive bias for efficient multi-hop reasoning.*

3.3 Causal patches on reasoning chains

Section 3.2 reveals an ordered trajectory of linearly decodable bridge entities. However, decodability alone does not show that the model uses these representations in subsequent recurrent steps. A probe may recover information that correlates with the computation without contributing to it. We therefore intervene on the decoded state and test whether the remaining recurrent computation follows the intervention.

Where should we intervene? A causal intervention first requires identifying where each intermediate entity is represented. Per-token probes provide a clear answer: when a bridge entity e_j becomes decodable in the sequence-level probes, its signal is concentrated at the position of the preceding relation r_j, whose application produces e_j. We therefore intervene at this position, where the model most strongly represents the current frontier entity. For each intervention, we pair a *base* chain, whose hidden state will be modified, with a *donor* chain, which provides the replacement.

Controlled frontier patching setup. We use the same model as in Section 3.2, fixing inference to six loops and restricting the evaluation to held-out ten-hop chains for a more controlled analysis. At the beginning of each recurrence, we use the entity probes to determine how far the computation has progressed. We consider a bridge entity *ready* when the correct entity is ranked top–1 and has positive calibrated pointwise $\mathcal{V}$-information. The *frontier* is then defined as the longest contiguous prefix of decoded entities. For each recurrence, we identify the modal frontier on the validation data and retain only held-out base–donor pairs that share this frontier. This fixes the computational stage before any test outcomes are observed. The selected frontiers are 1, 3, 5, and 8 for recurrences 2, 3, 4, and 5, respectively. We exclude recurrence 6 because most chains have already been completed by that point. Base and donor queries begin with different entities but share the same relation suffix.

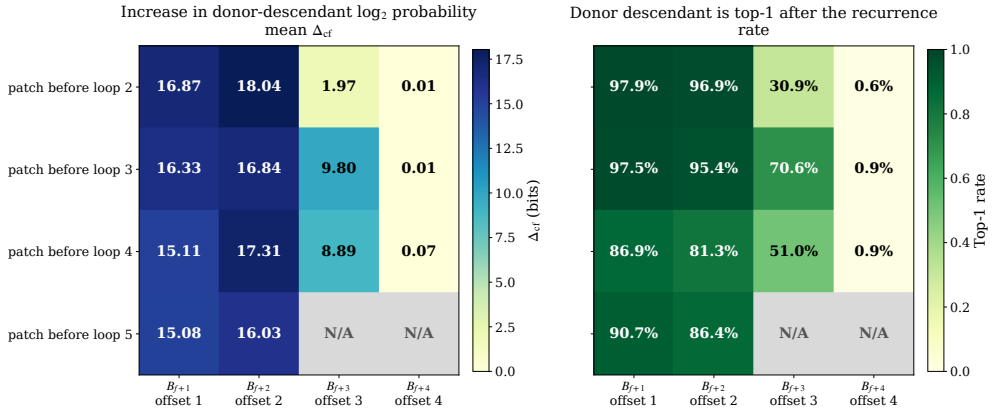


Figure 4: **One-recurrence causal reach after patching at the frontier.** Patches reliably propagate to nearby counterfactual descendants, with reach varying by recurrent state.

At the frontier position p, we replace only the component of the base residual that lies in the row space of the entity probe (specifically, the probe trained to decode the entity from the corresponding relation-token position): $h'_p = h_p + P_{\text{entity}}\left(h_p^{\text{donor}} - h_p\right)$, where P_{entity} is the projector onto that row space. We then continue the recurrent computation from the patched state. Because the relation suffix remains unchanged, this intervention tests a precise counterfactual: whether replacing the current frontier entity causes the model to follow the donor chain and propagate its descendants.

The patched entity propagates to its successors. We first examine the intervention's effect after a single additional recurrence. Using the frozen probes, we measure the increase in log-probability assigned to each donor-implied descendant, restricting the analysis to positions strictly downstream of the patched token. As shown in Figure 4, one recurrence propagates the patched entity through the next two, and sometimes three, bridge positions. Across intervention recurrences, descendants at offsets one and two gain between 15.1 and 18.0 bits of probe log-probability and become top–1 with rates of 81.3–97.9%. Propagation to the third descendant is more state-dependent, ranging from 2.0 to 9.8 bits, while the fourth descendant is essentially unaffected. All descendants are measured simultaneously after the same recurrence, showing that one application of the shared block can advance the counterfactual chain by multiple hops.

Patches steer the model's answer. Probe effects do not necessarily persist to the model's output. We therefore propagate each patched state through all remaining recurrences and evaluate the final answer. Entity-subspace patches at recurrences 1 to 5 produce the endpoint implied by the donor frontier with 79.07–98.57% accuracy (Figure 7, Appendix B.2). At recurrences 2-5, this accuracy exceeds that of the random-subspace control by 67.5-81.6 percentage points, showing that the probe identifies a causally relevant entity subspace. In contrast, patching from a different context that reaches the same frontier entity preserves the original answer with 99.57–99.87% accuracy.

The counterfactual answer generally emerges around recurrence 5 rather than immediately after the intervention, indicating that subsequent computation consumes the patched state instead of merely decoding it directly. A patch at recurrence 6 has no effect, consistent with the ten-hop computation already being complete. Together, the ordered decodability observed in Section 3.2 and these interventions support a causal account of the latent trajectory: productive applications of the shared block repeatedly advance an entity-conditioned frontier. This result demonstrates that looped models can implement multihop computation through causally active intermediate states.

Takeaway: *Multi-hop reasoning unfolds as a sequence of latent bridge entities, which become decodable in logical order across recurrent depth. Each entity serves as the key for retrieving its successor, and causal patches to the current frontier redirect both downstream entities and the model's final answer.*

4 Related work

Factual memorization and capacity. Allen-Zhu & Li (2023) introduce the synthetic-biography setting and establish an approximately two-bits-per-parameter memorization ceiling for Transformers. Zhu et al. (2025) extend this analysis to looped models, finding that recurrence does not increase storage capacity and suggesting that its benefits instead arise from improved knowledge manipulation. These studies measure knowledge after the forward pass is complete.

Recurrent depth and information accessibility. Recent work has established looped Transformers as a viable way to increase computational depth while reusing a fixed set of parameters (Zhu et al., 2025). These models can improve reasoning and generalization as the shared block is applied repeatedly (Saunshi et al., 2025; Kohli et al., 2026), but most studies evaluate only their final outputs. We instead examine how information evolves across recurrences of a frozen model. Because the native output head may be misaligned with intermediate representations, we sometimes turn to using tuned lens (Belrose et al., 2023) and linear probes (Alain & Bengio, 2018)to access information encoded in the hidden state which is not already accessible to the model's own readout

Latent computation in multihop reasoning. Kohli et al. (2026) show that recurrent depth improves systematic generalization and depth extrapolation on implicit multihop tasks, including a staircase relationship between chain length and the recurrences required for a correct endpoint. On a similar setting, Zhang et al. (2026) use causal patching to show that context-invariant bridge-entity representations support two-hop generalization. We extend this analysis to longer recurrent chains, tracing and causally intervening on multiple successive bridge entities. Complementarily, Fu et al. (2026) identify a related accessibility problem: a bridge entity may be decodable after one recurrence while remaining geometrically misaligned with the representation expected by the next.

5 Limitations and conclusions

Limitations. Our representational analyses focus on linearly accessible information: the tuned lens and entity probes are affine, while Equation 1 measures second-order geometry. Some information encoded nonlinearly may therefore go undetected. The single-hop experiments use one training run per configuration, three memorization loads, small models, and synthetic biographies with uniform attribute pools. These settings allow us to isolate retrieval dynamics and the causal role of intermediate entities, but limit the generality of our conclusions. Extending the analysis to looped models trained on natural text would test whether the same mechanisms persist at scale, although uncontrolled knowledge sources and contextual cues would make the signal harder to isolate.

Conclusions. Recurrence does not change what a model stores, but it changes when and how that knowledge becomes usable. In single-hop retrieval, facts become readable progressively across depth, and increasing memorization load shifts retrieval later without changing the final amount recovered. This schedule reflects a transition from a prompt-derived retrieval key to the corresponding stored value, with the value sometimes becoming decodable before the model's native output head can express it. In multi-hop reasoning, this process repeats: intermediate entities emerge in logical order, and each retrieved value becomes the key for the next step. Dynamic recurrence further promotes faster state turnover and earlier solvability, and causal interventions show that these representations are not merely probe-readable correlates, since changing the current frontier redirects downstream bridge entities and ultimately the model's answer. Together, these findings separate storage capacity from computational availability and show that recurrent depth organizes stored knowledge into a sequence of causally active intermediate states.

These findings suggest several practical principles for looped transformers. First, varying the number of recurrences during training appears especially promising, as it preserves generalization while encouraging earlier solutions and more efficient reuse of latent state. Second, increasing the number of training loops without intermediate-loop supervision spreads retrieval across greater depth rather than increasing capacity. This should be considered when selecting the training recurrence budget, as additional loops alone are not a substitute for model capacity, and intermediate-loop supervision may be necessary when earlier retrieval is desired. Finally, early-exit mechanisms should account for memorization load, as the required exit depth increases with the number of stored facts. Because facts can become internally decodable before appearing in the native logits, purely logit-based exit rules may also detect readiness late.

AI USE STATEMENT

We used a large language model as a writing and analysis aid: for prose editing of sections drafted by the authors, and for exploratory analysis scripts over result files that the authors then verified. Research questions, experimental design, model training, and all reported measurements are the authors' own. Every number in the paper was recomputed from the logged outputs by an author, and all LLM-assisted code was read and tested before use. We take responsibility for the final content of this work.

ETHICS STATEMENT

This work uses language models trained on synthetic data. No human subjects, personal data, or scraped text are involved, and the individuals studied do not exist. We see no direct pathway to harm from the measurements reported here.

REPRODUCIBILITY STATEMENT

Sections 2.1 and 3.1 specify the model architectures, recurrence budgets, datasets, and training objectives. We fully describe the construction, fitting, and evaluation of the linear lenses and intermediate-entity probes, including all data splits and masking procedures. Equation 1 defines the variance-ratio analysis and its permutation normalization, while Section 3.3 details the intervention protocol. Code and checkpoints will be released.

REFERENCES

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2018. URL https://arxiv.org/abs/1610.01644.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.1, knowledge storage and extraction. *arXiv preprint arXiv:2309.14316*, 2023.

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

Hengyu Fu, Tianyu Guo, Zixuan Wang, Hanlin Zhu, Jason D Lee, Jiantao Jiao, Stuart Russell, and Song Mei. Discoloop: Looping discrete embeddings and continuous hidden states for multi-hop reasoning. *arXiv preprint arXiv:2607.00341*, 2026.

Harsh Kohli, Srinivasan Parthasarathy, Huan Sun, and Yuekun Yao. Loop, think, & generalize: Implicit reasoning in recurrent-depth transformers. *arXiv preprint arXiv:2604.07822*, 2026.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers, 2025. URL https://arxiv.org/abs/2502.17416.

Zili Zhang, Yilin Wang, Heng Wang, Herun Wan, and Minnan Luo. Why knowing both hops is not enough: Understanding two-hop generalization in language models, 2026. URL https://arxiv.org/abs/2608.07261.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025.

A STORAGE CAPACITY OF OUR CHECKPOINTS

Figure 5 reproduces Figure 6 of Zhu et al. (2025) and shows highly similar results.

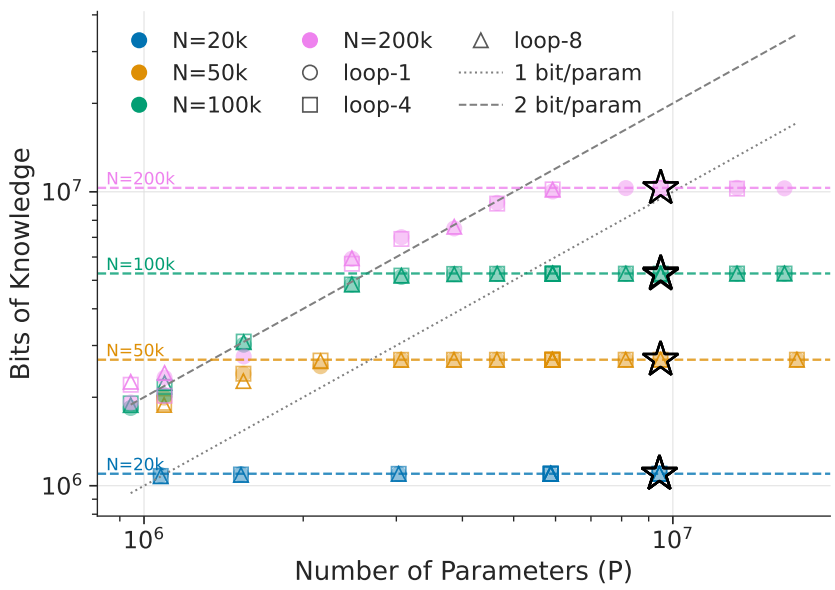


Figure 5: **Looping does not change what is stored.** Total memorised bits against parameter count for models trained on BioS, colours by dataset size and marker shapes by loop budget. Dashed horizontal lines are the ceiling $R_{\max}$ for each dataset. Loop-1, loop-4 and loop-8 coincide everywhere. Stars mark the models used throughout the paper: identical architecture across dataset sizes, all at their ceiling.

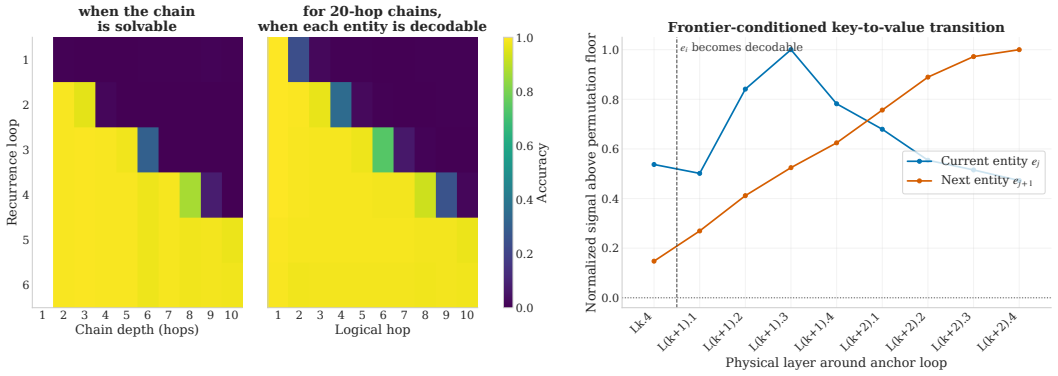


Figure 6: **(Left)** For R6, intermediate-entity decodability follows the same staircase as endpoint accuracy. **(Right)** For R6, the permutation-normalized MANOVA metric reveals a repeated key-to-value transition: after e_j becomes decodable, its signal declines while that of e_{j+1} grows.

B Further details of the multi-hop experiment

This appendix provides complementary analyses and additional details for the multi-hop experiments presented in Section 3.

B.1 Information retrieval in multi-hop chains

Figure 6 supports the claim that retrieval proceeds from keys to values, as discussed in Section 3.2. It also shows that each recurrence advances the reasoning chain at the same speed as hops become solvable.

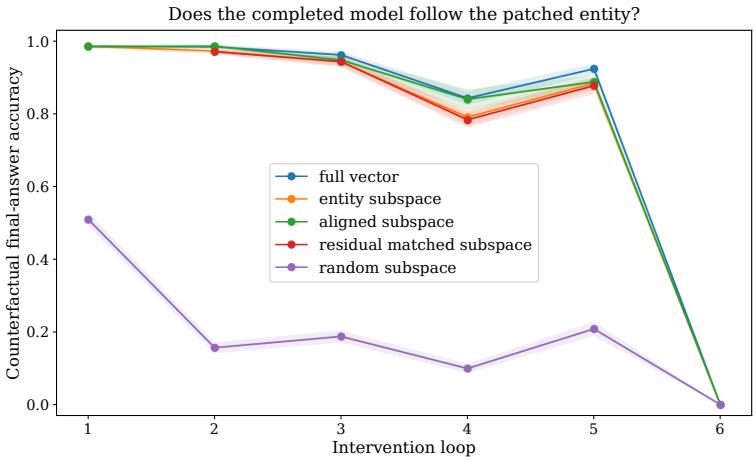


Figure 7: Final counterfactual-answer accuracy after patching at each recurrence and running the remaining loops. The model's native LM head follows entity-subspace interventions, while random-subspace patches are far less effective.

B.2 Causal patches on reasoning chains

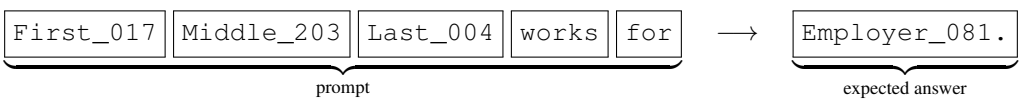
Figure 7 shows whether the model follows the patched entity until the end of the chain under different intervention methods. In particular, we intervene at each loop using the following methods:

- **Full vector:** We replace the entire residual vector with the corresponding donor vector.
- **Entity subspace:** We transfer only the donor-base difference projected onto the entity subspace (as in Section 3.3).
- **Aligned subspace:** We insert a validation-derived target-entity representation aligned across loops.
- **Residual-matched subspace:** We use a target-entity donor selected to minimize differences outside the entity subspace.
- **Random subspace:** We transfer an equal-dimensional, norm-matched random projection as a control.

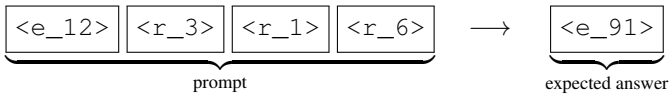
We observe that entity-subspace patching, which is the intervention used in our main experiments, performs as well as full-vector replacement. The counterfactual-answer accuracy remains above 79% for all intervention loops. As expected, intervening at loop 6 is ineffective because the computation has already finished. Finally, an equal-dimensional random- subspace intervention performs substantially worse, showing that the entity subspace contains information that is both predictive of entity identity and causally useful for counterfactual steering.

C Example Prompts

Single-hop retrieval. A biography prefix prompts the model to retrieve one stored attribute:



Multi-hop reasoning. A starting entity and sequence of relations prompt the model to compose stored atomic facts:



Here, the corresponding latent chain is <e_12> → <e_47> → <e_8> → <e_91>.