

DiscoLoop: Looping Discrete Embeddings and Continuous Hidden States for Multi-hop Reasoning

Hengyu Fu^{1*}Tianyu Guo^{1*}Zixuan Wang^{1,2*}Hanlin Zhu^{1*}Jason D. Lee¹Jiantao Jiao¹Stuart Russell¹Song Mei¹

Abstract

Large language models achieve strong performance on many reasoning tasks when allowed to externalize intermediate steps as Chain-of-Thought (CoT). However, many questions require the model to internalize the *multi-step reasoning* within a single forward pass before generating the answer. We study this challenge through *two-hop reasoning*, a representative task where the model must compose multiple pieces of parametric knowledge within a single forward pass. Standard non-recurrent Transformers suffer from a *depth-local storage* problem: facts learned in earlier layers are unavailable where second-hop retrieval happens. We found that Looped Transformers mitigate this issue by reusing the same memory, but still generalize imperfectly. We show that the remaining bottleneck is *representational*. In the two-hop reasoning task, the first loop often makes the correct bridge entity nearly perfectly decodable, yet the corresponding hidden state remains poorly aligned with the bridge token embedding. Surprisingly, an easy training-free realignment intervention nearly closes the generalization gap. Building upon this insight, we propose **DiscoLoop**, a looping architecture whose recurrence carries both a *discrete* embedding channel and a *continuous* hidden-state channel. DiscoLoop achieves near-perfect accuracy with substantially fewer training steps across symbolic and synthetic-language multi-hop reasoning tasks. When applied to real-world pretraining, DiscoLoop attains lower training loss and stronger benchmark performance than looped-transformer baselines, suggesting that the mixed-channel design transfers to practical language modeling.

1 Introduction

Current large language models (LLMs) heavily rely on long traces of Chain-of-Thought (CoT) [Wei et al., 2022] to achieve strong performance in various challenging tasks, including math [Jaech et al., 2024, Guo et al., 2025] and coding [Chen et al., 2021, Cao et al., 2026]. Yet many questions implicitly require several reasoning steps before generating the answer token, which must happen *internally* through the forward pass [Yang et al., 2024]. Ideally, an intelligent model should reason in a parameter and context-efficient way by *storing* atomic facts and *composing* them implicitly on demand, which reduces the need to verbalize every intermediate step in the context. Unfortunately, many recent works show that transformer-based LLMs struggle with such *implicit reasoning* [Ye et al., 2024, Press et al., 2023, Yang et al., 2025] despite their excellent performance with CoT.

Towards understanding the failure on implicit reasoning, a symbolic *two-hop reasoning* task [Wang et al., 2024, Ye et al., 2025b] was proposed as the most fundamental version of implicit reasoning. As an example, with **atomic facts** like “*Alice’s son is Bob*” and “*Bob’s wife is Carol*”, the **composite**

*Equal Contribution.

¹University of California, Berkeley. Email: {hengyuf, tianyu_guo, hanlinzhu}@berkeley.edu

²Princeton University. Email: zw2814@princeton.edu

query should be “*Who is the wife of Alice’s son?*”. Traditional transformer-based LLMs surprisingly struggle to learn this seemingly simple task [Yang et al., 2024, Biran et al., 2024, Balesni et al., 2024, Wang et al., 2024], and models often fail to compose two learned facts when the exact composite query was not observed during training.

The failure was previously attributed to a **depth-local storage** problem in standard transformers [Biran et al., 2024, Wang et al., 2024]. In this view, a non-recurrent Transformer solves two-hop reasoning through a depth-wise circuit by assigning different roles to layers of different depth. In the example above, earlier layers can recover the first-hop bridge entity, “*Bob*”, from “*Alice’s son is Bob*”, while later layers must use this bridge to retrieve the second-hop answer “*Carol*”, from “*Bob’s wife is Carol*”. This depth-wise division of labor requires the second-hop fact to be available in the later layers where the answer is retrieved. However, if that fact is observed during training only as an atomic fact and never in a compositional second-hop role, training provides little direct pressure to make it available at the required depth.

This perspective motivates looped Transformers [Wang et al., 2024, Zeng et al., 2025, Zhu et al., 2025c, Kohli et al., 2026] as a natural architectural fix. By applying the same Transformer block recurrently, a looped model reuses the same learned parametric memory across reasoning steps: the first loop can recover the bridge, and the next loop can query the same memory to resolve the second hop. However, despite substantially outperforming non-looped Transformers, existing looped-Transformer architectures can still suffer from low training efficiency [Wang et al., 2024] and imperfect generalization [Ye et al., 2025b] especially in out-of-distribution (OOD) settings, where the constituent atomic facts are observed during training only as standalone facts and never appear in any compositional two-hop training example [Ye et al., 2025b]. This raises our motivating questions:

If looping makes the same memory available across hops, why do vanilla looped transformers still fail to compose atomic facts? How do we resolve the remaining issue?

Based on the two questions, we design a controlled symbolic two-hop reasoning task and conduct mechanistic interpretability analyses on vanilla looped transformers (see Section 2 and Section 3). We identify a new **representation** bottleneck: after the first loop, the bridge entity is often decodable from the continuous hidden vector, yet this vector remains noisy and geometrically misaligned with the clean discrete embedding of the same entity that the next loop would ideally consume. We further show that a training-free realignment intervention nearly closes the performance gap in both ID and the more challenging OOD setting, which reveals a simple design principle: the recurrence should pass not only continuous hidden states, but also discrete embedding-aligned vectors across loops.

Motivated by this principle, we propose **DiscoLoop**, a looped Transformer architecture that keeps the memory-reuse benefit of recurrence while adding a decoded embedding channel to reduce loop-wise representation mismatch. DiscoLoop achieves near-perfect test accuracy with substantially fewer training steps across symbolic and synthetic-language multi-hop reasoning tasks. Moreover, on real-world pretraining tasks, DiscoLoop attains lower training loss and stronger performance on different benchmarks compared to looped-transformer baselines, suggesting that our mixed-channel design generalizes to realistic pretraining setups (see Section 5).

1.1 Related work

Out-of-context and multi-hop reasoning. Out-of-context reasoning (OCR) refers to models’ capability to answer a question where certain key information is missing from the context but learned during training. Previous works show that LLMs can achieve strong OCR performance in certain tasks like factual recall and implication [Feng et al., 2024, Huang et al., 2025], while they struggle with other OCR tasks, such as the reversal curse [Berglund et al., 2023, Allen-Zhu and Li, 2023, Zhu et al., 2024], in-weight multi-hop reasoning [Yang et al., 2024, Biran et al., 2024, Balesni et al., 2024], etc. We focus on boosting in-weight multi-hop reasoning capability in this work, which is one of the most important OCR tasks. Prior work [Wang et al., 2024] shows that transformers can learn two-hop reasoning through grokking (i.e., an extended period of training far beyond overfitting) when the test data is in distribution (ID), but totally fail for out-of-distribution (OOD) test data (See the rigorous definition of ID and OOD in Section 2). Biran et al. [2024], Wang et al. [2024] interpreted the failure reason as a depth-local storage issue. Ye et al. [2025a] showed that the explicit decodability of intermediate entities is insufficient for implicit reasoning, and second-hop generalization requires seeing the relevant atomic fact in the same compositional role during training.

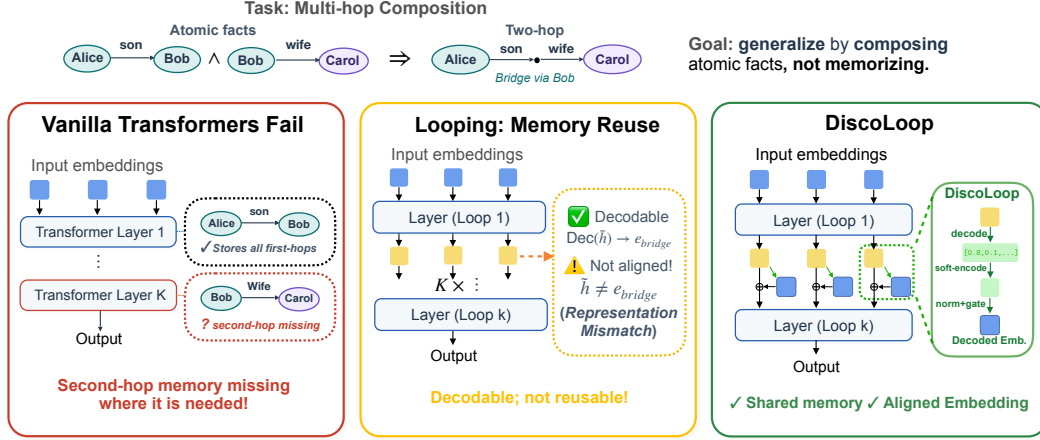


Figure 1: **Overview of our analysis and DiscoLoop.** The figure summarizes the two-hop reasoning setup, the depth-local storage issue of vanilla non-looped transformers, the representation-mismatch issue that remains in vanilla looped transformers, and our DiscoLoop architecture that injects a soft decoded embedding into the recurrent residual stream. The rigorous two-hop task formulation is defined in Section 2, and the DiscoLoop recurrence is formalized in Section 4.

Latent reasoning and looped transformer. Unlike explicit CoT [Wei et al., 2022], latent space reasoning moves the reasoning procedure from the textual token space to a latent space, including discrete latent space [Goyal et al., 2023, Wang et al., 2023, Cheng and Van Durme, 2024, Pfau et al., 2024, Su et al., 2025], continuous latent space [Hao et al., 2024, Zhu et al., 2025b,a, Gozeten et al., 2025, Zhang et al., 2025, Butt et al., 2025], looped transformer [Saunshi et al., 2025, Zeng et al., 2025, Zhu et al., 2025c, Kohli et al., 2026], etc. In particular, the looping architecture naturally resolves the storage issue of multi-hop reasoning, and the OOD test accuracy increases from 0 to a non-trivial number [Zeng et al., 2025, Zhu et al., 2025c, Kohli et al., 2026] compared to vanilla transformers. However, the OOD test accuracy is still far from perfect. Concurrent to our work, Kohli et al. [2026] studies implicit in-weight multi-hop reasoning with vanilla looped transformers. Their test split holds out 5% of atomic facts from training compositions within a single knowledge graph. In contrast, our OOD split uses a separate entity-disjoint graph of comparable size to the training graph and withholds all of its compositions from training, making the two-hop generalization more challenging in both scale and separation.

2 Task Formulation: Multi-hop Reasoning

Two-hop reasoning. We formalize a symbolic abstraction of the two-hop reasoning example in Section 1. Given a knowledge graph $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{F})$ with entity set $\mathcal{E}$, relation set $\mathcal{R}$, and edge set $\mathcal{F} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, an *atomic fact* is a triple $(a, r_1, b) \in \mathcal{F}$, i.e., an edge of $\mathcal{G}$ (e.g., $a = \text{Alice}$, $r_1 = \text{spouse}$, $b = \text{Bob}$). A *two-hop question* is a chain (a, r_1, r_2) (e.g., $r_2 = \text{workplace}$) whose answer c (e.g., San Francisco) is determined by composing two atomic facts: there exists a unique $b \in \mathcal{E}$ such that $(a, r_1, b) \in \mathcal{F}$ and $(b, r_2, c) \in \mathcal{F}$. We refer to b as the *bridge entity*. Equivalently, two-hop questions are generated by the inference rule

$$\forall a, b, c \in \mathcal{E}, \forall r_1, r_2 \in \mathcal{R}, \quad (a, r_1, b) \in \mathcal{F} \wedge (b, r_2, c) \in \mathcal{F} \implies (a, r_1, r_2, c). \quad (1)$$

The construction extends to k -hop questions $(a, r_1, \dots, r_k)$ for any $k > 1$, but $k = 2$ is the most canonical case and thus our main focus throughout the paper. Atomic facts and two-hop questions are presented as separate training examples, and the model is trained to predict the answer c directly from (a, r_1, r_2) without generating the bridge entity b . The model is therefore required to store the atomic facts *in weights* and to perform the composition *implicitly* within a single forward pass.

Two-graph design for ID/OOD splits. Beyond memorization, we expect that the trained models are able to generalize to compose facts, not only in in-distribution (ID) but also in out-of-distribution (OOD) test sets. We consider two *entity-disjoint* knowledge graphs $\mathcal{G}_A$ and $\mathcal{G}_B$ that *share the same relation vocabulary*. We designate $\mathcal{G}_A$ as the *in-distribution* graph: all atomic facts and a subset of

two-hop questions on $\mathcal{G}_A$ are used for training, while the remaining two-hop questions on $\mathcal{G}_A$ form the ID test set `test_id`. In contrast, $\mathcal{G}_B$ serves as the *out-of-distribution* graph, on which we train *only* its atomic facts; all two-hop questions on $\mathcal{G}_B$ are reserved as the OOD test set `test_ood`. The crucial asymmetry is that although the model has observed *every* atomic edge of $\mathcal{G}_B$ during training, it has *never* seen a two-hop composition over $\mathcal{G}_B$. Together, the two splits probe whether the composition rule learned on $\mathcal{G}_A$ transfers to held-out questions on the same graph (`test_id`) and, more stringently, to a disjoint entity set whose atomic facts the model has only seen in isolation (`test_ood`). This two-graph separation follows the protocol of Wang et al. [2024] and makes `test_ood` a stricter test of systematic generalization: unlike held-out compositions on $\mathcal{G}_A$, no fact in $\mathcal{G}_B$ has ever appeared as part of a compositional training example, so success requires applying the learned composition rule to facts seen only as standalone facts.

3 Looping Fixes Storage, but Not Representation

In this section, we train a vanilla looped transformer on the two-hop reasoning task of Section 2. We observe that while looping unlocks implicit two-hop generalization, the test accuracy remains far from perfect, especially out-of-distribution. We then conduct a mechanistic analysis to localize the failure as representational, and find that a simple training-free intervention on the hidden states at the bridge position is enough to nearly close the OOD gap.

3.1 Training setup

Dataset design. We instantiate the two-graph construction of Section 2 with $|\mathcal{E}_A| = |\mathcal{E}_B| = 500$ entities per graph and a shared relation vocabulary of size $|\mathcal{R}| = 50$. Every entity has exactly 10 outgoing edges, each labeled by a distinct relation, so each graph contains 5,000 atomic facts and 50,000 two-hop chains. During training, each entity and each relation is assigned its own dedicated symbolic token (e.g. `<a>`, `<r_1>`), so an atomic fact is rendered as a three-token sequence `<a> <r_1> <b>`, and a two-hop question as a four-token sequence `<a> <r_1> <r_2> <c>`. In both cases, the model is trained to predict the final token. The training set comprises *all* 10,000 atomic facts from both graphs (`train_atom`), together with 10,000 two-hop questions (`train_id`) drawn uniformly from $\mathcal{G}_A$. We hold out a further 2,000 $\mathcal{G}_A$ chains as the ID test set `test_id`, and sample 2,000 two-hop questions from $\mathcal{G}_B$ as the OOD test set `test_ood`.

Looped Transformer. As a starting point, we train a decoder-only Transformer from scratch in a *looping* pipeline. Let f_θ denote the stack of transformer blocks, and let $\mathbf{W} \in \mathbb{R}^{V \times d}$ be the (tied) embedding matrix, where V is the vocabulary size and d the hidden dimension. Given an input token sequence $x = (x_0, \dots, x_{T-1})$, we form the initial hidden states $\mathbf{H}^{(0)} = \mathbf{W}[x] \in \mathbb{R}^{T \times d}$, and then apply f_θ recurrently by feeding each iterate’s output back in as its own input:

$$\mathbf{H}^{(k+1)} = f_\theta(\mathbf{H}^{(k)}), \quad k = 0, 1, \dots, K-1, \quad (2)$$

The next-token logits are read off the final iterate through the same embedding matrix, $\ell = \mathbf{H}^{(K)} \mathbf{W}^\top$. We set $K = 2$ to match the maximum reasoning depth in the data, giving the model exactly two passes to compose two atomic facts. Intuitively, on a two-hop input (a, r_1, r_2) , we expect f_θ to retrieve the first-hop fact (a, r_1, b) in the first loop and reveal the bridge entity b in the residual stream, and then to use b together with r_2 in the second loop to retrieve (b, r_2, c) , obtaining the final answer c .

3.2 Results and mechanistic analysis

We now train the vanilla looped transformer described in Section 3.1 on the symbolic dataset and evaluate it across the four splits introduced there. We report accuracy as the fraction of inputs on which the model’s argmax-decoded next token at the answer position matches the ground-truth target.

Looped transformers do compose, but imperfectly. After 3,000 epochs of training, the looped transformer fits the training set perfectly on both atomic facts and two-hop chains (Table 1a). The key diagnostic is the intermediate *Stage-1* accuracy, obtained by reading the next-token prediction off the post-first-loop hidden states $\mathbf{H}^{(1)}$ via the LM head $\mathbf{W}$, i.e., applying f_θ only once ($K = 1$) at inference time. Stage-1 atomic accuracy is 100%, confirming that atomic facts are perfectly recalled from the weights within a single loop. In contrast, Stage-1 two-hop accuracy on `train_id` is only

8.8%, far below its perfect Stage-2 (final) score. The model therefore does *not* memorize two-hop answers as shortcuts inside f_θ ; instead, the second loop is what performs the composition, exactly as the $K = 2$ design intends. Despite this correctly decoupled behavior, the final accuracy is still imperfect: ID test accuracy plateaus at 71.1%, and OOD test accuracy at only 8.3%. In conclusion, the vanilla looped transformer does the right thing structurally, but its compositional reasoning capability is still far from perfect, particularly outside the training distribution.

A training-free intervention closes the ID/OOD gap. A looped transformer can in principle resolve the depth-local storage issue: the first loop may retrieve the bridge entity before the second-hop fact is needed. However, the second loop must still consume the first-loop representation of this bridge. We therefore test whether making this intermediate representation more embedding-like improves two-hop generalization.

We apply a targeted, training-free intervention illustrated in the left panel of Figure 2. Between the first and second loops, at position 1 only, we replace the hidden state with a convex combination of itself and the normalized embedding of the highest-probability decoded token,

$$\mathbf{H}_1^{(1)} \leftarrow (1 - \alpha) \mathbf{H}_1^{(1)} + \alpha \text{Norm}(\mathbf{W}[b_{\max}]), \quad \alpha \in [0, 1], \quad (3)$$

where $b_{\max} = \arg \max(\mathbf{W}\mathbf{H}_1^{(1)})$ is the top decoded token. Sweeping α produces the right panel: a small intervention ($\alpha = 0.1$) already lifts OOD accuracy from 8.3% to 25.9%, and at $\alpha \approx 0.5$ both ID and OOD accuracy approach 100%, achieving more than a *ten-fold* gain on OOD accuracy.

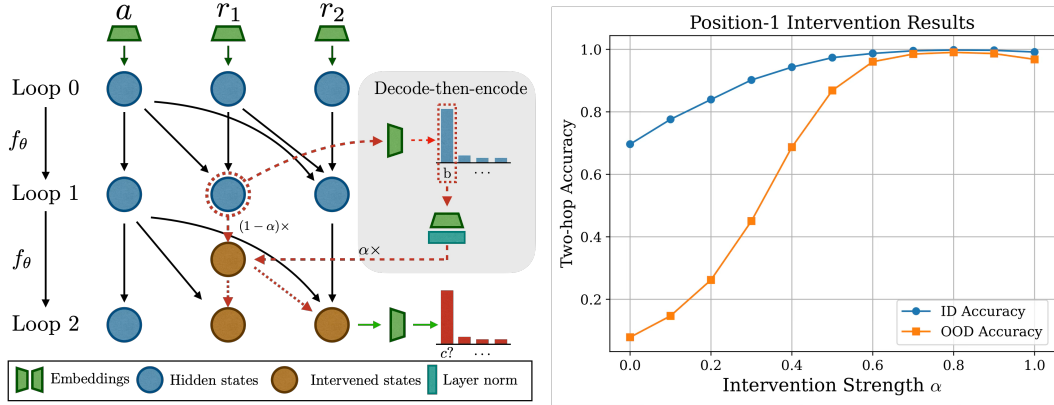


Figure 2: **Left:** the intervention procedure. At $\alpha = 0$ the position-1 representation is unchanged; at $\alpha = 1$ it is replaced entirely by the decoded bridge embedding $\text{Norm}(\mathbf{W}[b_{\max}])$. The substitution is applied only at position 1 and only between the two loops; the rest of the forward pass is untouched. **Right:** ID and OOD two-hop accuracy as a function of the intervention strength α in Eq. (3).

Decodable bridge, misaligned representation. To understand why this intervention helps, we apply a logit lens analysis to the position where the bridge entity b is expected to appear. For every two-hop test input (a, r_1, r_2) , we look up the ground-truth bridge b and examine the intermediate hidden state $\mathbf{H}_1^{(1)}$ at position 1 (the position of r_1). We report the intermediate Stage-1 LM-head probability $P(b \mid \mathbf{H}_1^{(1)}) = \text{softmax}(\mathbf{W}\mathbf{H}_1^{(1)})_b$ assigned to the ground-truth bridge token, and the cosine similarity between $\mathbf{H}_1^{(1)}$ and the bridge embedding $\mathbf{W}[b]$.

As shown in Table 1b, the model assigns probability essentially 1 to the correct bridge on both the ID and OOD splits, which indicates that the shared loop block has both pieces needed for composition: the first loop recovers the bridge, and the same block contains the parametric memory to answer (b, r_2) queries in the second loop. This suggests that the remaining failure cannot be simply explained by the depth-local storage issue that limits non-looped Transformers: the bridge is already available before the second loop, and the second-hop fact is ideally retrievable in the second loop since the same block has learned to recall all atomic facts within a single loop.

However, the hidden vector itself is not fully aligned with the bridge embedding direction. The cosine similarity stays around 0.3, and the OOD cosine is noticeably lower than the ID one. Thus, the second

Table 1: Results of a vanilla looped transformer on the symbolic dataset. *Stage 1* reads the next-token prediction off the post-first-loop hidden states $\mathbf{H}^{(1)}$; *Stage 2* is the model’s actual prediction after both loops.

(a) Training and test accuracy on the four splits.					(b) Logit lens at position 1, post-first-loop.		
	train_atom	train_id	test_id	test_ood		$P(b \mid \mathbf{H}_1^{(1)})$	$\cos(\mathbf{H}_1^{(1)}, \mathbf{W}[b])$
Stage 1	100.0%	8.8%	0.9%	0.0%	test_id	1.000	0.327
Stage 2	100.0%	100.0%	71.1%	8.3%	test_ood	1.000	0.266

loop receives a noisy continuous mixture rather than a clean copy of the discrete embedding $\mathbf{W}[b]$. Equivalently, the base model f_θ is asked to consume two qualitatively different input distributions across the two loops: clean token embeddings $\mathbf{W}[x]$ in the first loop, but noisy hidden states $\mathbf{H}^{(1)}$ in the second.

This suggests that the representation mismatch between the noisy continuous hidden state and the clean discrete embedding is a dominant cause of the generalization gap. The larger gap in the OOD case is consistent with this view: OOD entities appear only as endpoints of atomic facts during training and never as intermediate bridges, so f_θ is never directly pressured to align their post-first-loop hidden states with their embeddings. Therefore, Looping resolves the depth-local storage issue but leaves a representation bottleneck, and mixing in the clean embedding direction at the bridge position is enough to recover near-perfect accuracy. This motivates the architectural change we develop in the next section.

4 Architecture Design

The intervention in Section 3.2 suggests a simple design principle: between loops, the residual stream should carry not just the continuous hidden state computed by f_θ , but also a clean discrete embedding read off through the LM head. The targeted intervention realizes this idea fixed at one position and via a non-differentiable arg max. We now lift it into a learned, fully differentiable architecture that applies the same principle jointly at every position and every loop.

We propose **DiscoLoop**, a looping architecture that augments the residual stream with both *discrete* embeddings and *continuous* hidden states across the *loop* recurrence. Given a base model f_θ , we replace the vanilla recurrence (2) by

$$\mathbf{H}^{(k+1)} = f_\theta(\tilde{\mathbf{H}}^{(k)}), \quad \tilde{\mathbf{H}}^{(k+1)} = \mathbf{H}^{(k+1)} + \alpha^{(k)} \odot \text{RMSNorm}(\Phi(\mathbf{H}^{(k+1)})), \quad (4)$$

for $k = 0, \dots, K-1$, with $\tilde{\mathbf{H}}^{(0)} = \mathbf{W}[x] \in \mathbb{R}^{T \times d}$. Here $\odot$ denotes row-wise multiplication, so $\alpha^{(k)} = [\alpha_1^{(k)}, \dots, \alpha_T^{(k)}]$ is a token-wise gate along the sequence. Φ is a soft *decode-then-encode* operator applied independently at each position:

$$\Phi(h) = \sum_{v=1}^V p_v(h) \mathbf{W}[v], \quad p_v(h) = \frac{\exp((\mathbf{W}h)_v/\tau)}{\sum_{v'=1}^V \exp((\mathbf{W}h)_{v'}/\tau)}, \quad (5)$$

with temperature $\tau > 0$. Intuitively, $\Phi(h)$ is a weighted sum of token embeddings whose weights are the tempered next-token probabilities obtained by reading the LM head off h , which is the differentiable analogue of the hard substitution $\mathbf{W}[v_{\max}]$ used in Section 3.2. Adding $\alpha^{(k)} \odot \text{RMSNorm}(\Phi(\mathbf{H}^{(k+1)}))$ to the residual stream therefore injects a *discrete embedding channel* alongside the continuous hidden states, with a gate $\alpha^{(k)}$ controlling the injection strength.

Gate parametrization. We set $\alpha^{(K-1)} \equiv \mathbf{0}_T$, so the final prediction is still read directly from $\mathbf{H}^{(K)}$ by the LM head. For $k = 0, \dots, K-2$, we consider two parametrizations of $\alpha^{(k)}$. We can either set a *fixed* $\alpha^{(k)} \equiv \alpha^* \mathbf{1}_T$ with $\alpha^* > 0$ for all k , or design a *learnable* $\alpha^{(k)}$ by introducing a token-wise gate through a sigmoid activation σ :

$$\alpha_t^{(k)} = \sigma(\langle \mathbf{w}_\alpha, \Phi(\mathbf{H}_t^{(k+1)}) \rangle + b_\alpha), \quad \mathbf{w}_\alpha \in \mathbb{R}^d, b_\alpha \in \mathbb{R}. \quad (6)$$

The gate is conditioned on the decoded content $\Phi(\mathbf{H}_t^{(k+1)})$ at each token, so the model can inject the discrete embedding more strongly at positions where the decoded distribution is informative and turn the injection off elsewhere. The learnable variant introduces only $d + 1$ extra parameters in total (a single vector $\mathbf{w}_\alpha$ and bias b_α shared across loops and positions), which is negligible relative to the size of f_θ .

5 Experiments

We evaluate DiscoLoop in three settings of increasing realism. Section 5.1 first revisits the symbolic two-graph task from Section 3.1. Section 5.2 then studies a synthetic natural-language version of the same task in two question formats, testing whether the same effect persists beyond pure symbolic tokenization. Section 5.3 evaluates practical language modeling, asking whether the DiscoLoop also improves standard pretraining beyond the controlled multi-hop datasets.

5.1 Symbolic two-graph dataset

We re-use the symbolic dataset and training setup of Section 3.1. We compare DiscoLoop with two baselines: (i) a *vanilla looped transformer* of Eq. (2), which shares the same base model f_θ as DiscoLoop and differs only in whether the discrete embedding channel is added, and (ii) a *non-looped transformer* obtained by stacking K copies of f_θ as untied layers. This non-looped baseline matches the FLOPs of the looped models while strictly upper-bounding their parameter count. For DiscoLoop, we use a fixed gate $\alpha^* = 1$ and temperature $\tau = 1$. All other training hyperparameters are kept identical across the three models.

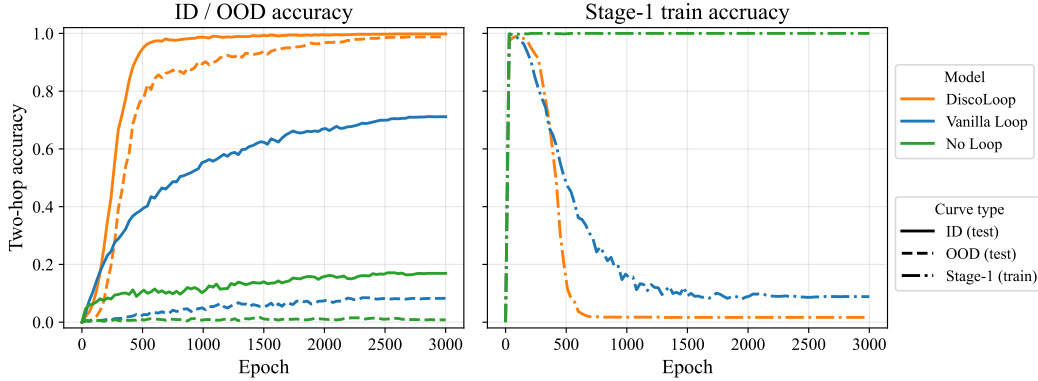


Figure 3: Two-hop accuracy along training on the symbolic dataset for the non-looped transformer, the vanilla looped transformer, and DiscoLoop. **Left:** Final two-hop accuracy on the held-out ID test set (solid) and the OOD test set (dashed). **Right:** Stage-1 two-hop training accuracy, obtained by reading the LM head after the first loop. Since the non-looped transformer has no recurrent stages, we report its final two-hop training accuracy as a reference.

DiscoLoop dominates both baselines. According to the left panel of Figure 3, DiscoLoop reaches near-perfect accuracy on both the ID and OOD test sets by the end of training, while the vanilla looped transformer plateaus at roughly 70% ID and below 10% OOD. The non-looped baseline performs the worst despite using $K \times$ parameters: only below 20% ID accuracy essentially 0% OOD. DiscoLoop also demonstrates substantially better training efficiency: we observe a **sharp phase transition** around epoch 500, after which both ID and OOD test accuracy already exceed 80%.

Stronger inductive bias towards compositional reasoning. The right panel of Figure 3 plots the Stage-1 two-hop *training* accuracy, obtained by reading the LM head off the post-first-loop hidden states $\mathbf{H}^{(1)}$ as described in Section 3.2. Early in training, both looped models behave like the non-looped baseline: Stage-1 training accuracy quickly hits 100% while ID and OOD test accuracy stay near zero, indicating that two-hop answers are being memorized directly in the weights of f_θ . During the phase transition, the Stage-1 training accuracy *collapses toward zero* just as the test

accuracies *jump up*. Once the model can no longer answer two-hop training questions from a single loop, it is forced to offload composition to the second loop. This is the looping architecture’s inductive bias against single-loop memorization shortcuts, leading to a more generalizable solution.

Crucially, the Stage-1 collapse is *earlier and sharper* for DiscoLoop than for the vanilla looped baseline, and its test accuracy rises earlier and converges higher. By injecting a cleaner signal via the discrete embedding channel Φ , DiscoLoop strengthens this inductive bias, pushing the model toward looping-based composition more aggressively, thus accelerating the phase transition and improving final generalization.

Beyond two-hop. The benefits of DiscoLoop are not limited to two-hop reasoning. We additionally evaluate it on a *three-hop* extension of the symbolic dataset and find that it again converges to near-perfect ID accuracy and substantial OOD generalization, while the vanilla looped transformer and the non-looped baseline both fail completely on the three-hop OOD split. Due to space constraints, we defer the full setup and results to Appendix A.2.

5.2 Synthetic natural-language dataset

To check that the architectural finding is not an artifact of symbolic tokenization, we re-run the same task on a synthetic natural-language version where atomic facts and two-hop questions are verbalized in English. We again generate a two-graph dataset as described in Section 2. Instead of a pure symbolic sequence, each atomic fact is rendered as “Finley’s teacher is Anya”, and we consider two natural verbalizations of the two-hop query that differ in the order of the relation chain:

- *Direct*: “Finley’s teacher’s wife is Charlie”.
- *Reverse*: “Who is the wife of the teacher of Finley?
Answer: Charlie”.

The two formats share the same atomic facts and the same answer for every two-hop chain; only the surface form of the question differs. The reverse format is closer to natural-language QA but requires the model to compose relations in the opposite order to that in which they appear in the input.

We again use $K = 2$ loops for both DiscoLoop and the vanilla looped transformer, and the corresponding non-looped baseline stacks two copies of f_θ as untied layers. We train all models for 5000 epochs, with the remaining training setup identical to the symbolic experiments. We use a learnable gate α for DiscoLoop on both synthetic-language datasets.

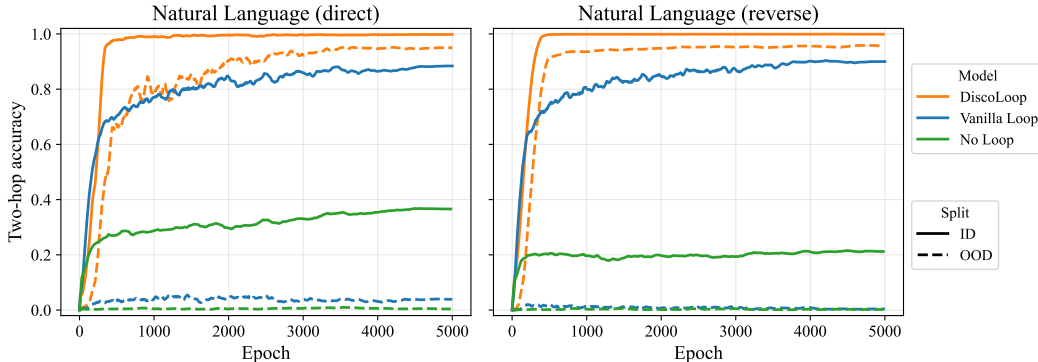


Figure 4: Two-hop accuracy on the two synthetic-language datasets (*direct* and *reverse* formats), comparing the non-looped transformer, the vanilla looped transformer, and DiscoLoop on the ID test set (solid) and the OOD test set (dashed). All models are trained under the same optimization recipe used in the symbolic setting, except that the total number of training epochs is 5000 rather than 3000.

Figure 4 shows that the language-setting curves mirror the symbolic case. For both formats, DiscoLoop outperforms both baselines in accuracy and training efficiency. DiscoLoop reaches near 100% ID accuracy and around 95% OOD accuracy, while the vanilla looped transformer reaches around 90% ID accuracy but fails OOD, especially in the reverse format. The non-looped baseline reaches only around 40% ID accuracy on the direct format and 20% on the reverse format, and

Table 2: Zero-shot evaluation scores for the 440M pretraining runs using the backbone architecture of Zhu et al. [2025c], with 24 layers, loop step 4, and tied input-output embeddings. For multiple-choice tasks, we report length-normalized accuracy to reduce length bias. Thus ARC-C, ARC-E, HellaSwag, PIQA, and SciQ use normalized accuracy; LAMBADA and RACE use accuracy. Bold indicates the best score for each benchmark. DiscoLoop is best or tied on six of the seven benchmarks.

Setting	ARC-C	ARC-E	HellaSwag	LAMBADA	PIQA	RACE	SciQ	Avg.
Vanilla loop	30.5	57.2	44.2	35.1	67.4	31.0	79.5	49.3
PonderLM	31.7	55.6	43.9	38.0	67.7	31.2	80.5	49.8
DiscoLoop (Ours)	32.6	57.6	44.2	37.6	68.4	31.3	81.5	50.5

essentially 0% OOD accuracy on both. DiscoLoop also generalizes perfectly on the ID test set well before the end of training, also around epoch 500, making it substantially more training-efficient than the baselines. We observe the same inductive-bias phenomenon as in the symbolic case, which we visualize in Appendix A.3.

5.3 Language modeling

We next test whether the benefits of DiscoLoop extend beyond controlled synthetic tasks to practical language modeling. Besides the vanilla loop baseline, we compare with PonderLM [Zeng et al., 2025], a related recurrent architecture that also maps intermediate predictions back into embedding space. The key difference is how the recurrent signal is carried: PonderLM maintains the recurrence only in embedding space, while DiscoLoop preserves the continuous hidden state and injects the decoded embedding as an additional discrete channel. Thus this comparison tests whether keeping both channels improves over both hidden-state-only looping and embedding-only looping.

We pretrain three 440M models on 20B tokens using a 6:4 mixture of FineWeb-Edu and FineMath [Niklaus and Kydlíček, 2026, Allal et al., 2025]. The FineMath component makes the corpus more math- and reasoning-heavy than pure web text. All models share the same model backbone as Zhu et al. [2025c] with hidden dimension $d = 1024$, 24 layers, loop step 4, and tied input-output embeddings. We keep the training recipe fixed and vary only the recurrent computation mechanism. For a controlled comparison, all recurrent models use four total applications of the backbone. Additional training details are provided in Appendix B.

DiscoLoop improves the language modeling. Table 2 reports zero-shot evaluation on seven standard language-modeling benchmarks. DiscoLoop obtains the best average score, improving over the vanilla Loop baseline from 49.3 to 50.5, and also outperforming PonderLM’s average score of 49.8. DiscoLoop is best or tied on six of the seven benchmarks, with the largest numerical improvements over Loop on ARC-C, LAMBADA, and SciQ. PonderLM remains strongest on LAMBADA. When aggregating across benchmarks, DiscoLoop achieves the highest average score, suggesting that the looped reasoning mechanism yields broad gains rather than improvements confined to a small subset of tasks. This indicates that the mechanism does not merely solve the synthetic compositional task, but also transfers to standard language model pretraining. We also present the training loss curves in Appendix B. DiscoLoop surpasses the vanilla loop after training on roughly 13B tokens and maintains this advantage through the end of training.

6 Conclusion and Discussion

We have studied implicit in-weight multi-hop reasoning through the lens of looped transformers. Our contribution has been two-fold. First, we have provided a mechanistic explanation for why vanilla looped transformers do not fully solve the task. Although the bridge entity is often already decodable after the first loop, the hidden state carrying it is not well aligned with the clean token embedding that the next loop would naturally consume. A simple training-free intervention that injects the decoded bridge embedding has nearly closed the ID/OOD gap, showing that representation mismatch is a central bottleneck. Second, motivated by this finding, we have proposed DiscoLoop, which augments the loop recurrence with both a continuous hidden-state channel and a decoded discrete-embedding channel. DiscoLoop has achieved strong performance on symbolic and synthetic-language two-hop

reasoning tasks, improving both training efficiency and OOD generalization, and has also shown gains in our open-domain pretraining experiments.

Looking forward, it would be interesting to test whether DiscoLoop length-generalizes: trained on questions with at most k hops, can it extrapolate to questions with more than k hops? This would be a stronger test of whether the architecture learns a more generalizable compositional procedure rather than a depth-specific strategy. Moreover, our pretraining experiments have been limited to moderate-scale models because of compute constraints, so larger-scale verification remains important. Beyond training looped models from scratch, an important next step is to study whether the same looping recipe can be introduced through continual pretraining of larger pretrained models. This would test whether existing pretrained non-looped models can acquire the discrete-continuous recurrence without full retraining, and whether the benefits of DiscoLoop persist at more realistic model scales.

7 Acknowledgements

HF and JDL acknowledge support of NSF IIS 2107304, NSF CCF 2212262, NSF CAREER Award 2540142, NSF 2546544, NSF CCF 2019844 and ONR N00014-24-1-2639. SM acknowledges support of ONR N00014-24-1-2639.

References

- Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM2: When smol goes big – data-centric training of a small language model, 2025. URL <https://arxiv.org/abs/2502.02737>.
- Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.2, knowledge manipulation. *arXiv preprint arXiv:2309.14402*, 2023.
- Mikita Balesni, Tomasz Korbak, and Owain Evans. The two-hop curse: LLMs trained on $a \rightarrow b$, $b \rightarrow c$ fail to learn $a \rightarrow c$. 2024.
- Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The reversal curse: LLMs trained on "a is b" fail to learn "b is a". *arXiv preprint arXiv:2309.12288*, 2023.
- Eden Biran, Daniela Gottesman, Sohee Yang, Mor Geva, and Amir Globerson. Hopping too late: Exploring the limitations of large language models on multi-hop queries. *arXiv preprint arXiv:2406.12775*, 2024.
- Natasha Butt, Ariel Kwiatkowski, Ismail Labiad, Julia Kempe, and Yann Ollivier. Soft tokens, hard truths. *arXiv preprint arXiv:2509.19170*, 2025.
- Ruisheng Cao, Mouxiang Chen, Jiawei Chen, Zeyu Cui, Yunlong Feng, Binyuan Hui, Yuheng Jing, Kaixin Li, Mingze Li, Junyang Lin, et al. Qwen3-coder-next technical report. *arXiv preprint arXiv:2603.00729*, 2026.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*, 2024.
- Jiahai Feng, Stuart Russell, and Jacob Steinhardt. Extractive structures learned in pretraining enable generalization on finetuned facts. *arXiv preprint arXiv:2412.04614*, 2024.
- Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika,

- Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL <https://zenodo.org/records/12608602>.
- Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. *arXiv preprint arXiv:2310.02226*, 2023.
- Halil Alperen Gozeten, M Emrullah Ildiz, Xuechen Zhang, Hrayr Harutyunyan, Ankit Singh Rawat, and Samet Oymak. Continuous chain of thought enables parallel exploration and reasoning. *arXiv preprint arXiv:2505.23648*, 2025.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.
- Yixiao Huang, Hanlin Zhu, Tianyu Guo, Jiantao Jiao, Somayeh Sojoudi, Michael I Jordan, Stuart Russell, and Song Mei. Generalization or hallucination? understanding out-of-context reasoning in transformers. *arXiv preprint arXiv:2506.10887*, 2025.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.
- Harsh Kohli, Srinivasan Parthasarathy, Huan Sun, and Yuekun Yao. Loop, think, & generalize: Implicit reasoning in recurrent-depth transformers. *arXiv preprint arXiv:2604.07822*, 2026.
- Joel Niklaus and Hynek Kydlíček. Smoldata. <https://huggingface.co/collections/HuggingFaceFW/smol-data>, 2026.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
- Jacob Pfau, William Merrill, and Samuel R Bowman. Let’s think dot by dot: Hidden computation in transformer language models. *arXiv preprint arXiv:2404.15758*, 2024.
- Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, 2023.
- 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.
- Dijia Su, Hanlin Zhu, Yingchen Xu, Jiantao Jiao, Yuandong Tian, and Qingqing Zheng. Token assorted: Mixing latent and text tokens for improved language model reasoning. In *International Conference on Machine Learning*, pages 57144–57163. PMLR, 2025.
- Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. Grokked transformers are implicit reasoners: A mechanistic journey to the edge of generalization. *arXiv preprint arXiv:2405.15071*, 2024.
- Xinyi Wang, Lucas Caccia, Oleksiy Ostapenko, Xingdi Yuan, William Yang Wang, and Alessandro Sordani. Guiding language model reasoning with planning tokens. *arXiv preprint arXiv:2310.05707*, 2023.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface’s transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.
- Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. Do large language models latently perform multi-hop reasoning? *arXiv preprint arXiv:2402.16837*, 2024.
- Sohee Yang, Nora Kassner, Elena Gribovskaya, Sebastian Riedel, and Mor Geva. Do large language models perform latent multi-hop reasoning without exploiting shortcuts? In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 3971–3992, 2025.
- Jiaran Ye, Zijun Yao, Zhidian Huang, Liangming Pan, Jinxin Liu, Yushi Bai, Amy Xin, Weichuan Liu, Xiaoyin Che, Lei Hou, et al. How do transformers learn implicit reasoning? *arXiv preprint arXiv:2505.23653*, 2025a.
- Jiaran Ye, Zijun Yao, Zhidian Huang, Liangming Pan, Jinxin Liu, Yushi Bai, Amy Xin, Liu Weichuan, Xiaoyin Che, Lei Hou, et al. How does transformer learn implicit reasoning? *arXiv preprint arXiv:2505.23653*, 2025b.
- Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process. *arXiv preprint arXiv:2407.20311*, 2024.
- Boyi Zeng, Shixiang Song, Siyuan Huang, Yixuan Wang, He Li, Ziwei He, Xinbing Wang, Zhiyu Li, and Zhouhan Lin. Pretraining language models to ponder in continuous space. *arXiv preprint arXiv:2505.20674*, 2025.
- Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen, and Xin Eric Wang. Soft thinking: Unlocking the reasoning potential of llms in continuous concept space. *arXiv preprint arXiv:2505.15778*, 2025.
- Hanlin Zhu, Baihe Huang, Shaolun Zhang, Michael Jordan, Jiantao Jiao, Yuandong Tian, and Stuart Russell. Towards a theoretical understanding of the ‘reversal curse’ via training dynamics. *Advances in neural information processing systems*, 37:90473–90513, 2024.
- Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Emergence of superposition: Unveiling the training dynamics of chain of continuous thought. *arXiv preprint arXiv:2509.23365*, 2025a.
- Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. *arXiv preprint arXiv:2505.12514*, 2025b.
- 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*, 2025c.

A Additional experiments and training details on synthetic datasets

A.1 Training details

We implement the vanilla non-looped transformers, looped-transformers and DiscoLoop all through PyTorch and Huggingface Transformers [Paszke et al., 2019, Wolf et al., 2019]. All the synthetic experiments are done on 8 NVIDIA H200 GPUs within 168 hours at maximum.

A.2 Three-hop reasoning experiment

To check that the architectural advantage of DiscoLoop is not specific to two-hop composition, we run the same comparison on a three-hop extension of the symbolic two-graph task.

Three-hop questions. Following the symbolic abstraction in Section 2, a *three-hop question* is a chain (a, r_1, r_2, r_3) whose answer d is determined by composing three atomic facts: there exist unique $b, c \in \mathcal{E}$ such that $(a, r_1, b), (b, r_2, c), (c, r_3, d) \in \mathcal{F}$. We refer to b and c as the two bridge entities, and the three-hop input is rendered as a five-token sequence $\langle a \rangle \langle r_1 \rangle \langle r_2 \rangle \langle r_3 \rangle \langle d \rangle$, with the model trained to predict the final answer token.

Dataset. We re-use the two-graph construction of Section 3.1: $|\mathcal{E}_A| = |\mathcal{E}_B| = 500$ entities per graph, a shared relation vocabulary of size $|\mathcal{R}| = 50$, and out-degree 10, so each graph contains 5,000 atomic facts. The training set is composed of *all* 10,000 atomic facts from both graphs, 5,000 two-hop questions, and 5,000 three-hop questions, both sampled from $\mathcal{G}_A$. We evaluate on four held-out test splits of 1,000 chains each: two-hop ID and three-hop ID, sampled from $\mathcal{G}_A$ disjoint from training; and two-hop OOD and three-hop OOD, sampled from $\mathcal{G}_B$. Since no compositional question over $\mathcal{G}_B$ is ever seen in training, both OOD splits cleanly probe systematic compositional generalization at the corresponding reasoning depth.

Models and training. We set $K = 3$ loops for both the vanilla looped transformer and DiscoLoop, matching the maximum reasoning depth in the data. The non-looped baseline correspondingly stacks three copies of f_θ as untied layers. We use a fixed gating $\alpha = 1$ for DiscoLoop. All other hyperparameters (base model, optimizer, learning rate, batch size, schedule) follow Section 3.1.



Figure 5: Test accuracy on the three-hop symbolic dataset for the non-looped transformer, the vanilla looped transformer, and DiscoLoop. Solid lines: ID test set. Dashed lines: OOD test set. **Left:** two-hop test accuracy. **Right:** three-hop test accuracy.

Results. Figure 5 shows test accuracy on the four splits over training. The curves resemble the two-hop case but with a more pronounced separation. On the two-hop splits (left panel), DiscoLoop reaches near-perfect ID accuracy and around 90% OOD, while the vanilla looped transformer plateaus near 40% ID and 5% OOD, and the non-looped baseline fails on both. On the three-hop splits (right panel), DiscoLoop again converges to near-perfect ID accuracy and reaches roughly 65% OOD.

Strikingly, both the vanilla looped transformer and the non-looped baseline remain near zero on the three-hop OOD split throughout training.

In conclusion, DiscoLoop’s advantage is not specific to two-hop composition: the same mechanism continues to accelerate training and improve generalization at deeper reasoning depths. Moreover, the gap between DiscoLoop and the baselines *widens* with depth. The vanilla looped transformer can still partially solve two-hop ID, but almost fails on three-hop, where each additional hop requires another clean handoff between loops. This is consistent with the mechanistic story in Section 3.2: representational misalignment at the bridge position compounds across hops in the vanilla model, while DiscoLoop’s discrete channel suppresses it at every hop, unlocking compositional generalization that is otherwise unreachable.

A.3 Stage-1 inductive bias on the natural-language datasets

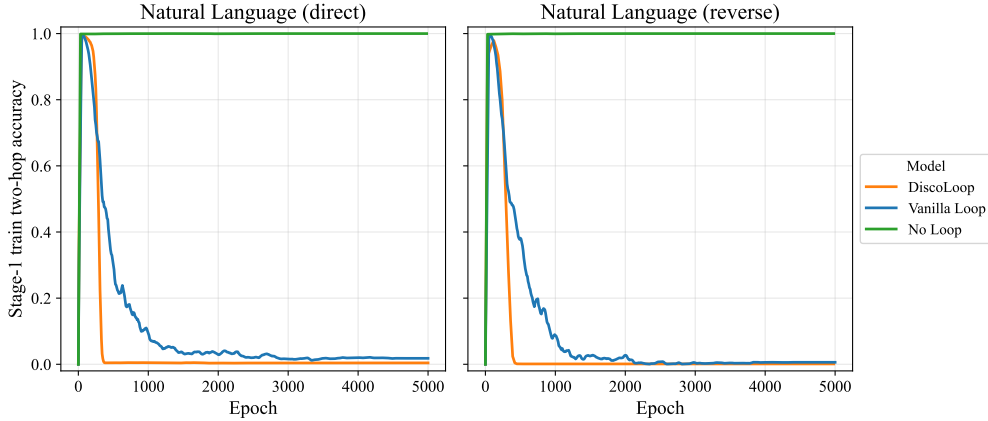


Figure 6: Stage-1 two-hop training accuracy on the two synthetic natural-language datasets (*direct* and *reverse* formats), for the non-looped transformer, the vanilla looped transformer, and DiscoLoop. As in the symbolic case, both looped models suppress their Stage-1 training accuracy as composition is offloaded from f_θ to the looping recurrence; the suppression is markedly sharper for DiscoLoop than for the vanilla looped transformer. The non-looped baseline retains $\sim 100\%$ throughout, having no second loop to offload composition to.

Figure 6 plots the Stage-1 two-hop training accuracy on the two natural-language datasets, computed exactly as in the symbolic case (Section 5.1). The pattern mirrors that case: both looped models initially fit the two-hop training set within a single loop and then sharply suppress this single-loop accuracy once composition is offloaded to the looping recurrence. The collapse is remarkably sharper for DiscoLoop than for the vanilla looped transformer—DiscoLoop drops to near-zero Stage-1 accuracy within roughly 400 epochs, while the vanilla baseline takes more than 2,000 epochs to do the same. The non-looped baseline retains $\sim 100\%$ Stage-1 accuracy throughout, since it has no second loop to offload composition to. The natural-language results therefore reproduce the same inductive-bias phenomenon observed in the symbolic setting, with DiscoLoop again strengthening it.

B Language modeling pretraining details

This section gives the full training and evaluation details for the open-domain language-modeling experiments in Section 5.3. We adopt the Ouro architecture from Zhu et al. [2025c] because it provides a strong and established design for loop transformers. The recorded launcher configuration corresponds to the vanilla loop baseline. DiscoLoop and PonderLM use the same architecture scale, data mixture, tokenizer, optimizer, sequence length, and distributed training setup, changing only the recurrent computation mechanism. All recurrent models apply the backbone four times, i.e., $K = 4$.

For DiscoLoop, we use a top- k approximation to the decode-then-encode operator Φ in Eq. (5). For a hidden state h , let

$$\mathcal{T}_k(h) = \text{TopK}(\{p_v(h)\}_{v=1}^V) \subseteq [V]$$

denote the set of k tokens with the largest probabilities under the LM-head distribution $p(h)$. We then replace Φ by

$$\Phi_k(h) = \sum_{v \in \mathcal{T}_k(h)} p_v(h) \mathbf{W}[v],$$

where we mask all logits outside the top- k set and then computing the embedding average on this truncated support. This reduces compute and memory overhead, since most probabilities $p_v(h)$ are negligible in practice. We find that $k = 128$ is sufficient to retain essentially all nontrivial probability mass in our pretraining runs.

For PonderLM, four backbone applications correspond to three ponder steps under the convention of the original PonderLM formulation, where the final backbone application is treated as the standard final forward pass rather than a ponder step.

Architecture. We use a looped transformer backbone Zhu et al. [2025c] with hidden size 1024, 24 layers, and 16 attention heads. The recurrent computation uses four total applications of the backbone. The context length is 8192, with maximum position embeddings set to 16384 and RoPE base $\theta = 10^6$. The model uses SwiGLU activations, tied input/output embeddings, no QKV bias, and vocabulary size 129,280.

Training data. Unless otherwise specified, all runs use the same two-source pretraining mixture: 60% FineWeb-Edu sample data and 40% FineMath-4plus. Both sources are loaded as streaming parquet datasets. We use the DeepSeek-V3-0324 tokenizer.

Optimization. Models are trained with AdamW and Muon enabled. We use learning rate 3×10^{-3} , AdamW $\epsilon = 10^{-15}$, Muon momentum 0.95, gradient clipping at norm 1.0, and a linear learning-rate schedule with 1000 warmup steps and final absolute learning rate 10^{-5} . The effective batch size is 128, the sequence length is 8192.

Token budget and distributed setup. The Loop baseline is trained with a target budget of 20B tokens. With 2 nodes, 8 GPUs per node, batch size 8, sequence length 8192, and gradient accumulation 1, each step processes 1,048,576 tokens. This gives 19,074 optimization steps, corresponding to 20,000,538,624 total training tokens.

Evaluation. We evaluate checkpoints using `lm-evaluation-harness` in the zero-shot setting. The reported benchmarks are ARC-C, ARC-E, HellaSwag, LAMBADA, PIQA, RACE, and SciQ. For multiple-choice tasks where `lm-evaluation-harness` [Gao et al., 2024] provides `acc_norm`, we report `acc_norm`; otherwise we report `acc`.

Training loss curves. Figure 7 shows the pretraining global average loss for the three settings: DiscoLoop, Vanilla loop, and PonderLM. Early in training, Vanilla loop has the lowest loss, while DiscoLoop and PonderLM start slightly higher. In the second half of training, however, DiscoLoop undergoes a clear transition and overtakes Vanilla loop, becoming the best-performing run in the late stage. In the zoomed region, this advantage is sustained and reaches roughly 10^{-2} in global average loss at its largest, while PonderLM remains above DiscoLoop throughout this phase.

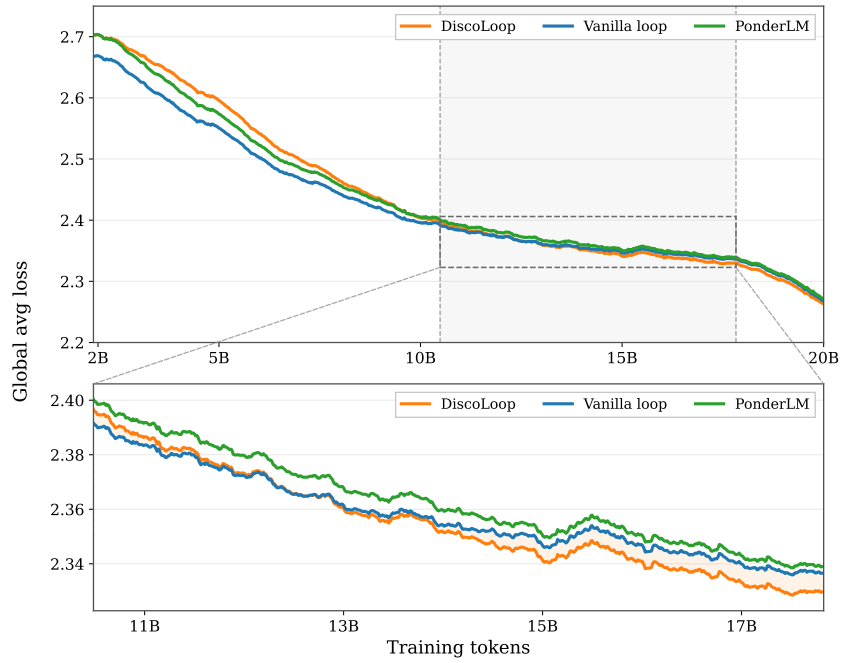


Figure 7: **Pretraining loss curves for the language modeling.** Global average loss versus training tokens for DiscoLoop, Vanilla loop, and PonderLM over the full 20B-token training budget (19,074 optimization steps). For readability, curves are smoothed with an exponential moving average. The upper panel shows the full training trajectory, while the lower panel zooms into the late-training regime from roughly 10B to 17B tokens; the dashed box and connector lines in the upper panel indicate the zoomed interval. Vanilla loop leads early in training, but DiscoLoop overtakes it in the second half of training and attains the lowest loss in the zoomed region, with a peak separation from Vanilla loop on the order of 10^{-2} . PonderLM also starts above Vanilla loop and, although it steadily improves, remains above DiscoLoop throughout the zoomed window.