LEARNING TO REASON MORE IMPLICITLY IMPROVES EXPLICIT REASONING ON HARDER PROBLEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) trained with chain-of-thought (CoT) on simpler problems often fail to generalize to harder problems that require more reasoning steps. Humans, by contrast, can learn to solve harder problems by practicing on simpler problems until they can perform known steps mentally. Inspired by this, we ask whether LLMs can also generalize to harder problems by learning to reason more implicitly, i.e., to be able to perform more steps without generating intermediate tokens, even when they still generate CoTs at test time. In this work, we propose step dropout (SeDo), a simple approach for teaching models to reason more implicitly. SeDo trains models on both complete CoTs and CoTs with randomly omitted steps. This teaches models individual steps while also encouraging them to implicitly infer missing CoT steps. Across five models and four tasks, our results show that SeDo both (1) substantially improves the learning efficiency of implicit multi-step reasoning, and (2) helps models generalize to harder problems. Surprisingly, SeDo models generalize better even when given complete CoT prefixes, indicating that implicit reasoning also teaches more generalizable skills, beyond robustness to incomplete steps. We further show implicit reasoning helps explicit reasoning in models not trained with SeDo: in looped Transformers, more loops, which allow more computation for implicit reasoning, monotonically improve single-step accuracy within a CoT, even on calculations the models already solve perfectly without CoT.

1 INTRODUCTION

Large language models (LLMs) have achieved impressive performance on mathematical and symbolic reasoning tasks, often through Chain-of-Thought (CoT; Nye et al., 2021; Wei et al., 2022). However, controlled studies show that models that solve simpler instances of a task often struggle with harder ones, which require applying similar skills over more steps (Anil et al., 2022; Shojaee et al., 2025; Malek et al., 2025). These failures raise the question of whether models can robustly use what they have learned (Zhou et al., 2024). This work studies how to improve generalization to harder problems when training only on simpler ones.

Similar to models learning from CoT, humans often begin learning to solve a class of problems by working through each step deliberately. However, humans do not stop there: with practice, they internalize familiar steps, which helps them solve new and harder problems that rely on these skills (Cooper & Sweller, 1987; Taatgen, 2013). Inspired by this learning process, we ask two questions: (1) how can models learn to reason more implicitly, i.e., perform more steps within each forward pass; and (2) does learning to reason more implicitly help models solve harder problems even when they can write out every reasoning step with CoTs?

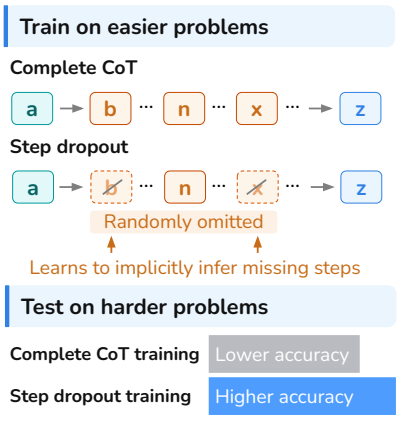


Figure 1: Step dropout randomly omits CoT steps during training, encouraging models to reason more implicitly. This improves generalization to harder problems while models still generate explicit CoTs at test time.

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.

Learning implicit reasoning is difficult: Yao et al. (2025) find that the data required for multi-hop reasoning grows exponentially with reasoning depth. However, they also find that this requirement drops substantially when models first learn fewer hops before more. Our key intuition is thus that *explicit CoT steps can serve as building blocks for multi-step implicit reasoning*. We further hypothesize that by jointly training on short and long prediction horizons derived from the same CoT, models can learn stronger implicit reasoning skills. To this end, we propose *step dropout* (SeDo), a simple approach for teaching models to reason more implicitly. As shown in Figure 1, SeDo simply omits random reasoning steps from CoTs with probability p and trains models jointly on both complete and partial CoTs. This way, the model learns to jointly predict both the next reasoning step, and more steps ahead, with partial previous steps available, which encourages implicit reasoning. We also study a variant, SeDo-R, which masks the omitted steps instead of removing them. We include more detailed explanations of both approaches in Appendix A.[1]

We first test whether joint prediction makes implicit reasoning easier to learn on DEPO (Allen-Zhu, 2025), a controlled multi-hop task. In DEPO, each instance provides a randomly generated directed cycle with its edges presented in random order. This makes DEPO a clean test of implicit reasoning, because models cannot succeed by simply skipping unnecessary steps. Across two architectures, joint prediction substantially improves implicit reasoning over answer-only training. SeDo also outperforms iCoT-SI (Deng et al., 2024), a method specifically developed to internalize CoT.

We further show that SeDo improves the performance of explicit reasoning on harder problems. Concretely, we evaluate SeDo and SeDo-R with three base models on three tasks: Maze (Su et al., 2025), Quickselect (Veličković et al., 2022), and GSM-∞ (Zhou et al., 2025). Trained on simpler problems, both methods generalize better than standard CoT training to problems that require more steps, and they generally outperform token-level removal and masking. For example, on harder Quickselect problems, SeDo improves Qwen3-1.7B's greedy accuracy from 24.5% to 41.5%. Moreover, SeDo still improves generalization when the next reasoning step is not unique, despite requiring more training data. SeDo's advantage also remains after further training with reinforcement learning, and improves generalization even when models are trained on their own generated CoTs.

Surprisingly, SeDo still generalizes better than standard CoT training when both models are given complete CoT prefixes, suggesting that it learns skills that generalize better. Linear probes further show that SeDo models do compute omitted intermediate values implicitly. Finally, we show that better implicit reasoning improves explicit reasoning beyond SeDo. In pretrained looped Transformers, more loops increase the implicit reasoning capacity of models and also improve single-step accuracy within a CoT, even on calculations the models already solve perfectly without CoT.

In summary, we jointly train models to predict the next step and steps further ahead in a CoT. This improves generalization even when models generate CoTs, suggesting that *learning to reason more implicitly, i.e., performing more steps internally, improves explicit reasoning on harder problems*.

2 Related work

Implicit reasoning. Recent work has studied whether and how language models reason implicitly, without producing intermediate tokens. Yang et al. (2024) show that models can internally recall and use intermediate entities to solve some multi-hop questions, and Nikankin et al. (2025) find that models use a bag of heuristics to solve arithmetic. However, multi-hop performance can be influenced by shortcuts such as entity co-occurrence. Controlling for these confounds, Yang et al. (2025b) find that models can still perform implicit reasoning, but success rates vary substantially across relations. Moreover, Yao et al. (2025) show in controlled settings that the amount of training data required for k-hop reasoning grows exponentially with k, though curriculum learning can substantially reduce this requirement.

Using CoT to learn implicit reasoning has also been studied. Some approaches distill explicit reasoning into implicit reasoning models. Deng et al. (2023) distill a teacher's CoT hidden states into a student model, while Yu et al. (2024) train on filtered input-answer pairs generated by CoT reasoning. Alternatively, Deng et al. (2024) progressively remove CoT tokens during training. Continuous-CoT

[1] Su et al. (2025) also studied omitting reasoning steps to switch between fast and slow thinking. However, here we show that it can teach models to internalize reasoning steps, and further show this helps models achieve better generalization. We include a more detailed discussion in §2.

methods instead feed the previous step's final hidden state back into the model, rather than the embedding of a decoded token (Hao et al., 2025; Shen et al., 2025). However, longer latent sequences can be unstable: to mitigate this, Wei et al. (2026) introduce step-level supervision to stabilize training. These works mostly aim to eliminate discrete intermediate tokens. In contrast, we train models to perform more reasoning internally while continuing to generate and use discrete CoT steps.

Learning with fewer reasoning tokens. Similar to SeDo, Su et al. (2025) drop structured parts of CoT traces and show improved performance both with and without intermediate reasoning tokens. However, they motivate trace dropping as learning pattern-based shortcuts, focusing on controllable fast and slow reasoning. We instead ask whether dropping steps teaches models to reason implicitly, and whether this improves explicit reasoning on harder problems. Answering these questions requires separating internalization from skipping unnecessary steps: we develop experiments on DEPO (§3) and linear probes (§5.1) accordingly. Moreover, Liu et al. (2024) iteratively train on generated shorter traces and observe generalization improvements. However, their warm-start procedure uses manually constructed shorter solutions, which often merge multiple steps into a single one instead of omitting them. Thus, fewer written steps need not mean less explicit reasoning. In contrast, by randomly omitting steps, we make them strictly absent. Importantly, we also show that predictions from complete CoT prefixes are more accurate (§5). Besides shortening traces, Chen et al. (2024) show that masking individual CoT tokens improves mathematical reasoning. In contrast, our focus on internalization motivates step-level masking and removal. We show that both substantially outperform token-level masking on harder problems (§4).

3 Multi-horizon prediction improves implicit reasoning

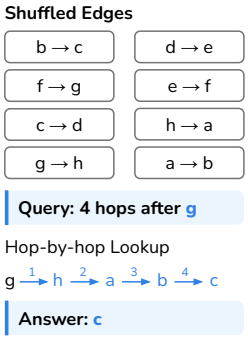
Learning implicit reasoning from final answers alone is difficult, because models need to learn multiple steps together without intermediate supervision (Wen et al., 2025; Altabaa et al., 2025; Hanneke et al., 2026). CoT makes individual steps easier to learn; however, because every preceding step is written out in the CoT, models never need to perform multiple steps implicitly. We therefore ask whether learning individual steps from CoT can help models learn longer implicit computations.

We expect models to learn longer implicit computations more easily by building on shorter ones. In line with this, Yao et al. (2025) show that a curriculum starting from fewer hops makes implicit multi-hop reasoning easier to learn, and prior work suggests that neural networks tend to reuse previously learned skills when acquiring new ones (Prakash et al., 2024; Du et al., 2025). We therefore train models to predict both the next CoT step and steps further ahead, since predicting k steps ahead requires performing the needed intermediate steps from the previous $k-1$ ones implicitly.

Step dropout (SeDo). Our main approach, SeDo, constructs partial CoTs by randomly removing reasoning steps and trains models on both complete and partial CoTs. Complete CoTs provide supervision for individual steps, while partial CoTs train models to infer omitted intermediate results needed for later predictions. This creates different prediction horizons without changing the architecture or the next-token objective. However, SeDo provides no supervision for the steps it removes. We therefore also study SeDo-R, which trains models to predict these steps and replaces each with a single mask token in the CoT prefix used to predict later steps. In this case, models learn to predict every step while still learning to reason without seeing all previous steps.

To test whether the improvement of joint prediction extends beyond SeDo, we also evaluate multi-token prediction (MTP; Gloeckle et al., 2024) and future-summary prediction (FSP; Mahajan et al., 2026) as alternative multi-horizon training approaches. MTP uses separate heads to predict future tokens, while FSP predicts a bag-of-words summary of a future token window. Unlike SeDo, both require additional prediction layers and keep the preceding CoT complete.

Experimental setup. To study implicit reasoning, we use DEPO, an in-context multi-hop reasoning task from Allen-Zhu (2025), illustrated in Figure 2. Each example provides a randomly generated directed cycle with its edges presented in random order. Given a starting node and a hop count k, the model needs to predict the node reached after k hops. Each intermediate node determines which edge to follow next, so the model *has to compute these nodes even when it does not write them out*. Therefore, DEPO tests whether models internalize reasoning rather than skipping unnec-



Method	Canon-Llama			GPT2-RoPE		
	K8	K12	K16	K8	K12	K16
Answer-only	**100.00**	98.83	62.50	0.00	0.78	0.78
Full-fade	86.72	21.09	0.39	14.06	0.78	0.00
MTP	**100.00**	99.22	94.92	**95.31**	50.00	10.55
FSP	**100.00**	99.61	80.08	26.95	0.39	0.78
iCoT-SI	98.44	99.22	84.77	75.00	3.91	0.39
SeDo	**100.00**	**100.00**	**98.05**	**95.31**	**71.09**	**37.11**
SeDo-R	**100.00**	99.61	96.48	80.47	2.34	1.17

Figure 2 (left). Illustration of the DEPO task. **Table 1 (right).** Direct-answer accuracy (%) on DEPO. Joint prediction methods (SeDo, SeDo-R, MTP, and FSP) are helpful for learning implicit reasoning, as they all substantially outperform the answer-only baseline.

essary steps. Concretely, we use the intermediate nodes as CoT supervision during training and evaluate the accuracy of predicting the final node directly without generating intermediate tokens.

Following Allen-Zhu (2025), we use an 8-layer Canon-Llama and a 12-layer GPT2-RoPE. We train both architectures from scratch for 150k steps on problems with up to 16 hops. Specifically, MTP and FSP both use a four-token prediction window and add three and one auxiliary layers, respectively. We further introduce two CoT-based baselines: Full-fade, which mixes complete CoTs and answer-only examples; and iCoT-SI (Deng et al., 2024), which progressively removes CoT tokens. For CoT-based methods except iCoT-SI, we gradually replace CoT examples with answer-only examples between 30k and 90k steps using a cosine schedule. We include more details in Appendix B.

Results. Table 1 shows that *joint prediction is generally helpful*: SeDo, MTP, and FSP all substantially improve over answer-only training on both architectures, with SeDo achieving the highest accuracy at every tested horizon. In particular, at 16 hops, SeDo improves accuracy over answer-only training from 62.50% to 98.05% on Canon-Llama and from 0.78% to 37.11% on GPT2-RoPE. It also outperforms iCoT-SI, which is specifically developed to internalize CoT steps through a staged curriculum. Moreover, SeDo learns longer horizons faster: on Canon-Llama, it reaches and stays above 80% accuracy at every tested horizon through 16 hops after 75k updates, compared with 145k for iCoT-SI (Figure 7). In contrast, Full-fade performs poorly at longer horizons despite also using CoT supervision. This suggests that learning individual steps from complete CoTs is not enough: training models to predict across multiple steps helps them use these computations implicitly.

4 Generalization of Explicit Reasoning to Harder Problems

We next ask whether learning to reason more implicitly helps explicit reasoning generalize. Concretely, we train on simpler instances and evaluate on instances requiring more reasoning steps, *while allowing models to generate a full CoT at test time*. We focus on SeDo and SeDo-R: neither requires architectural changes, so both can be applied directly to pretrained models; both train models to infer results omitted from earlier steps, which may help when their own generated CoTs are incomplete; and both naturally work with full reasoning steps rather than individual tokens, encouraging implicit computation across steps rather than merely completing the current step.

4.1 Generalization Across Tasks

Experimental setup. We evaluate on three tasks: two algorithmic tasks, Maze (Su et al., 2025) and Quickselect (Veličković et al., 2022), and a natural-language math reasoning task, GSM-∞ (Zhou et al., 2025). All three tasks have controllable complexity, measured by a task-specific count H. In Maze (Figure 3, left), given a maze, a start, and a goal, the model needs to generate a solution path. We use A* search events as the CoT for training, where H is the number of search events. In Quickselect (Figure 3, right), given an array and a rank, the model needs to find the element at that rank. We use the steps of the quickselect algorithm as the CoT, recording each partition action and the updated array, where H is the number of pivot comparisons. In GSM-∞ (Figure 5),

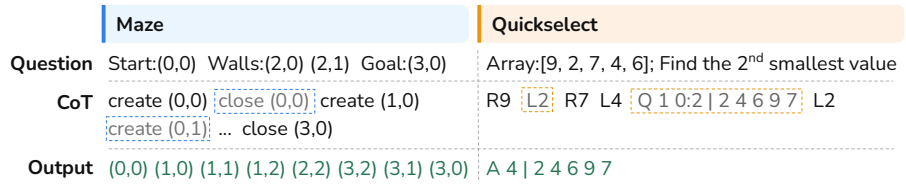


Figure 3: Simplified examples of Maze and Quickselect traces. Dashed boxes mark steps that SeDo may omit. In Maze, `create` and `close` are A* search events. In Quickselect, Q records the current state, L/R record partition decisions, and A gives the answer.

	Maze				Quickselect				GSM-∞			
H	$[200, 525)$		$[525, 650)$		$[140, 300)$		$[300, 380)$		$[26, 31)$		$[31, 36)$	
Method	p@1	p@16	p@1	p@16	p@1	p@16	p@1	p@16	p@1	p@16	p@1	p@16
Full	50.00	67.19	6.00	15.83	50.38	81.75	24.50	66.75	55.42	75.00	40.00	60.83
Token Drop	58.05	80.78	7.00	21.83	60.75	90.25	33.75	80.25	49.58	76.25	34.17	62.08
MFT	52.97	70.94	9.00	23.00	63.38	82.88	34.75	68.75	50.42	75.00	39.17	69.58
SeDo	**61.33**	**85.70**	11.83	**38.83**	68.75	**97.12**	41.50	**96.25**	**60.83**	**84.58**	**49.17**	**77.08**
SeDo-R	58.75	77.66	**12.67**	34.00	**70.00**	89.88	**43.75**	76.00	58.75	82.08	47.08	73.33

Table 2: Qwen3-1.7B test performance (%) on two harder ranges per task. p@1 is greedy accuracy; p@16 is pass at 16. SeDo and SeDo-R outperform Full across all evaluated ranges and metrics, and SeDo achieves the highest pass at 16 in every range.

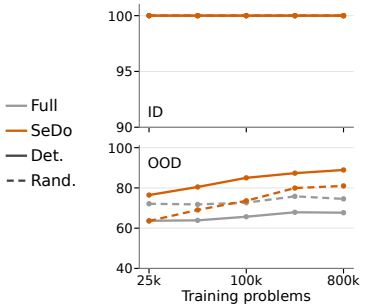
the model is prompted to solve math word problems. We use a CoT where each step computes one intermediate quantity, with H the number of mathematical operations. We train on simpler problems ($75 \leq H < 200$ for Maze, $60 \leq H < 140$ for Quickselect, and $16 \leq H < 26$ for GSM-∞) and test both within these ranges and on harder problems with larger H.

We post-train three base models on each task: Qwen3-0.6B-Base, Qwen3-1.7B-Base (Yang et al., 2025a), and Gemma3-1B-PT (Gemma Team, 2025). For SeDo and SeDo-R, we construct partial CoTs by omitting or masking each CoT step independently with probability 0.3. On Maze and Quickselect, half of the training CoTs are partial; on GSM-∞, each problem has one complete and three partial CoTs. The prompt and final answer are never omitted or masked.

We compare against three baselines: Full, which trains only on complete CoTs; Token Drop, which randomly omits individual tokens; and masked thought training (MFT; Chen et al., 2024), which replaces individual tokens with masks but still trains the model to predict the masked tokens. Token Drop and MFT are the token-level counterparts of SeDo and SeDo-R, respectively, allowing us to compare token-level and step-level training. We report both the greedy accuracy (p@1) and pass at k from sampled solutions (p@k). Further experimental details are provided in Appendix D.

Results. Table 2 shows results for Qwen3-1.7B: *SeDo and SeDo-R improve explicit reasoning on harder problems* across all three tasks, under both greedy decoding and sampling. For example, on Quickselect problems with $300 \leq H < 380$, SeDo improves greedy accuracy from 24.5% to 41.5% and p@16 from 66.8% to 96.3%. On Maze problems with $525 \leq H < 650$, p@16 increases from 15.8% to 38.8%. These gains also extend to natural-language reasoning: on GSM-∞ problems with $31 \leq H < 36$, SeDo improves greedy accuracy from 40.0% to 49.2% and p@16 from 60.8% to 77.1%. Moreover, the gains generally grow with problem difficulty: SeDo's p@16 gain over Full on Quickselect increases from 0.3 percentage points on ID problems to 15.4 and 29.5 points on the two OOD ranges, with similar trends on the other tasks and models. Appendix E reports results for all three models, including ID performance. Thus, learning to reason more implicitly improves generalization even when models can generate explicit reasoning steps.

Comparisons with Token Drop and MFT further show that *omitting full steps is more effective than omitting tokens*. SeDo achieves higher p@16 than both Token Drop and MFT in every OOD range across all three models. The difference is especially clear on GSM-∞, where each step contains



	Maze				Quickselect			
H	$[200, 525)$		$[525, 650)$		$[140, 300)$		$[300, 380)$	
Method	p@1	p@16	p@1	p@16	p@1	p@16	p@1	p@16
SFT								
Full	46.6	65.7	4.0	17.0	33.5	68.9	14.3	39.5
RL								
Full + GRPO	56.1	76.3	10.8	29.7	44.1	79.8	20.0	56.5
SeDo + GRPO	**65.6**	**90.4**	**20.5**	**55.7**	**76.9**	**96.8**	**59.0**	**92.5**
Self-generated CoT								
Full + Identity	69.6	82.0	22.5	38.3	57.1	85.4	25.0	59.8
Full + SeDo	**76.8**	**95.3**	**31.2**	**70.5**	**62.8**	**97.0**	**30.3**	**91.0**

Figure 4 (left). Scaling with deterministic (Det.) and randomized (Rand.) Maze traces: validation p@32 on ID ($100 \leq H < 200$) and OOD ($200 \leq H < 650$) mazes. **Table 3 (right).** Test accuracy (%) of Qwen3-0.6B after SFT, GRPO, or training on self-generated CoTs.

a longer calculation: for $31 \leq H < 36$, the p@16 scores of Token Drop and MFT reach 62.1% and 69.6%, compared with 77.1% for SeDo. Moreover, masking can improve greedy accuracy over dropping, but dropping benefits more from sampling. For example, on the harder Quickselect range, SeDo-R reaches 43.8% greedy accuracy versus 41.5% for SeDo, but its p@16 is 76.0% versus 96.3%. The token-level baselines show a similar trend on Quickselect: MFT can improve greedy accuracy over Token Drop, while Token Drop consistently achieves higher p@16.

4.2 Generalization with non-unique reasoning steps

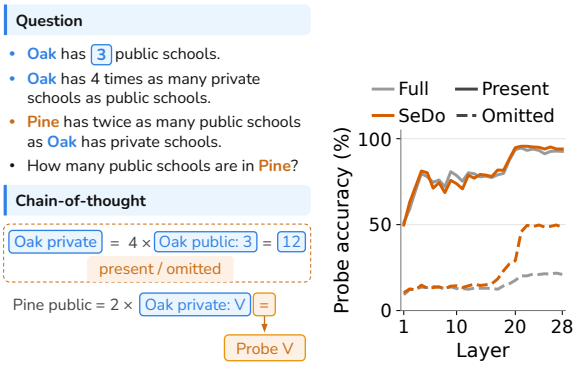
Does SeDo's generalization improvement require the next step to be uniquely determined? When several next steps are possible, the CoT prefix may not specify which omitted steps were taken, making them harder to infer implicitly. We therefore compare deterministic and randomized A* traces on the same mazes. The randomized version breaks ties between equally scored nodes at random and randomizes the order in which neighbors are added, so the same search history can have different valid continuations. We scale training from 25k to 800k mazes on Qwen3-0.6B, and increase the optimization budget proportionally.

SeDo improves generalization even with randomized traces, but requires more data and training. On ID instances, both methods achieve near-perfect performance with both trace types. In contrast, on OOD instances, with deterministic CoTs, SeDo outperforms Full at every data scale. However, with randomized CoTs, SeDo underperforms Full at small scales, performs similarly at 100k mazes, and outperforms it at 400k and 800k. Notably, SeDo's OOD performance consistently improves with scale, while Full improves more slowly and saturates early.

4.3 Generalization across training settings

So far, we have studied SeDo in SFT on externally provided CoTs. In practice, SFT is often followed by reinforcement learning (RL) to further improve reasoning (DeepSeek-AI et al., 2025). If Full has already learned correct solutions but rarely produces them, RL could make those solutions more likely (Yue et al., 2025) and close the gap with SeDo. Moreover, available training data may not come with clearly separated steps that SeDo can omit. One possible approach is to train on the available data first, then have the model generate its own solutions with identifiable steps and apply SeDo to those. Therefore, this section tests (1) whether SeDo's advantage remains after RL, and (2) whether SeDo improves generalization using the Full model's own CoTs, as a first step towards the use of self-generated data. We run these experiments with Qwen3-0.6B on Maze and Quickselect.

SeDo's advantage remains after RL. For both tasks, we start from the Full and SeDo SFT checkpoints and train with GRPO (Shao et al., 2024) without further step dropout. We train for 2,000 updates, with 16 problems per update and eight sampled solutions per problem. Both models already solve nearly all problems in the SFT range under sampling, so we extend the training range to $60 \leq H < 300$ for Quickselect and $75 \leq H < 525$ for Maze, with 32,000 problems per task.



	Incomplete		Complete		
	Full	SeDo	Full	SeDo	Gain (%)
Maze					
ID ($H < 200$)	15.4	**95.4**	96.9	**97.9**	+1.0
OOD $[200, 650)$	1.9	**46.6**	36.8	**46.5**	+26.4
Quickselect					
ID ($H < 140$)	4.3	**84.5**	100.0	100.0	0.0
OOD $[140, 380)$	3.5	**59.0**	82.9	**86.2**	+3.9
GSM-∞					
ID $[16, 26)$	62.7	**91.5**	98.5	**99.0**	+0.4
OOD $[26, 36)$	50.8	**77.9**	92.1	**94.0**	+2.0

Figure 5 (left). Probing the intermediate value V when its calculation is present or omitted: the linear probe predicts V's value at the equals sign at the bottom. The curves show Qwen3-1.7B's probing accuracy, where solid and dashed lines denote present and omitted calculations. **Table 4 (right).** Greedy accuracy of Qwen3-1.7B from complete and incomplete CoT prefixes. Gain is the relative improvement of SeDo over the Full model.

As Table 3 shows, *SeDo still generalizes better after RL*. For example, on Quickselect problems with $300 \leq H < 380$, SeDo reaches 59.0% greedy accuracy and 92.5% p@16, compared with 20.0% and 56.5% for Full. Moreover, on Maze problems with $525 \leq H < 650$, SeDo reaches 20.5% and 55.7%, roughly doubling Full's 10.8% and 29.7%.

SeDo also works with self-generated CoTs. We use the Full SFT model to solve the same 32,000 problems as in RL and keep only the correct, parsable solutions. Using the same collected solutions, we continue training two copies of the Full checkpoint for 20,000 updates. *Identity* uses complete solutions, whereas SeDo mixes complete and partially dropped solutions.

Table 3 shows *SeDo also improves generalization when trained on the model's own solutions*. On Quickselect problems with $300 \leq H < 380$, SeDo improves p@16 from 59.8% to 91.0% over Identity, and on Maze problems with $525 \leq H < 650$, from 38.3% to 70.5%. Greedy accuracy improves as well, from 25.0% to 30.3% on Quickselect and from 22.5% to 31.2% on Maze. We include the full experimental details of these two experiments in Appendix C.

5 Understanding the Benefits of Implicit Reasoning

This section studies how SeDo improves explicit reasoning. Having shown that SeDo models reason implicitly without CoT on DEPO (§3), we first examine whether they also implicitly compute omitted intermediate values when using CoTs. We then ask whether SeDo's generalization gains are limited to handling incomplete CoTs, or also extend to predictions from complete CoT prefixes. Finally, we test whether better implicit reasoning also improves explicit reasoning outside SeDo. For this, we study pretrained looped Transformers, where we can control the amount of implicit computation by varying the number of loops used.

5.1 Probing the Computation of Omitted Values

SeDo trains models to make predictions from incomplete CoTs. However, prediction accuracy alone does not fully show whether they are performing implicit reasoning: they might also take shortcuts to produce the answer without computing these values. We thus use linear probes to examine this.

Experimental setup. We generate new GSM-∞ problems within the training range to test implicit reasoning separately from generalization. For each problem, we select a variable V used later in the CoT and vary its inputs so that V takes values from 10 to 29. For each variant, we create two CoTs: one with V's line of calculation and one without it. We then train a linear probe on each layer to predict V from the hidden state where V is used to calculate the next variable, i.e., at the = sign (see Figure 5). For example, in `U = V * 2 =`, we probe at the second = to predict the value of V.

Results. *The probing results suggest that SeDo computes omitted values implicitly.* Figure 5 shows results for Qwen3-1.7B. When the calculation of V is present, probes predict its value accurately for both models. In contrast, when V's calculation is omitted, probe accuracy in layers 21–28 reaches 50% for SeDo, compared with 20% for Full, indicating SeDo's implicit reasoning. We observe similar patterns on Qwen3-0.6B and Gemma3-1B, and include more details in Appendix H.

5.2 Prediction from complete and incomplete CoT prefixes

Does learning to reason more implicitly only help SeDo models handle CoTs with missing steps, which may occur more often on harder problems? Or does it also help them reason more accurately when all preceding steps are complete and correct? To distinguish these possibilities, we compare SeDo and Full on the same correct CoT prefixes, either complete or with some steps omitted. If SeDo models also predict better from complete, correct prefixes, this would suggest that they learn more generalizable skills.

Experimental setup. We evaluate the Full and SeDo models on the original ID and OOD test problems. For each problem, we give the model a prefix of the ground-truth CoT and ask it to generate the next part of the solution: the solution path after the search events in Maze, a partition state at a randomly chosen position in Quickselect, and a calculation step in GSM-∞. We either keep the prefix complete or omit each preceding step independently with probability 0.3, and generate the continuation with greedy decoding. We report the valid-path accuracy in Maze, exact next-state accuracy (rank, active interval, and array) in Quickselect, and calculation accuracy in GSM-∞.

Results. *SeDo predicts better even from complete prefixes.* Table 4 shows results for Qwen3-1.7B. With complete prefixes, SeDo improves OOD accuracy from 36.8% to 46.5% in Maze, from 82.9% to 86.2% in Quickselect, and from 92.1% to 94.0% in GSM-∞. These gains are larger on OOD than on ID problems: in Maze, for example, the relative gain is only 1.0% on ID problems but 26.4% on OOD problems. As expected, the gap is much larger with incomplete prefixes: OOD accuracy increases from 1.9% to 46.6% in Maze and from 3.5% to 59.0% in Quickselect. Handling missing steps alone therefore cannot explain SeDo's gains, suggesting that learning to reason more implicitly also helps models learn more generalizable skills. We include more details and results in Appendix G, and observe the same pattern on Qwen3-0.6B and Gemma3-1B.

5.3 Implicit reasoning helps explicit reasoning beyond SeDo

We have shown that learning to reason more implicitly with SeDo improves the generalization of explicit reasoning. *We now ask whether this link extends beyond SeDo.* Suppose a model can already perform a calculation correctly when given the numbers directly: can it also perform the same calculation reliably within a CoT? If not, can improving its multi-step implicit reasoning make this CoT step more accurate? We study these questions using pretrained looped Transformers, where we can give the same model more computation by increasing the number of loops, without changing its parameters.

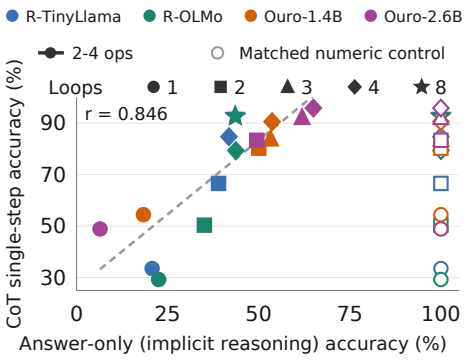


Figure 6: Models that solve multi-step arithmetic better also compute CoT steps more accurately. Hollow points show that the same single-step calculations are solved perfectly when the numbers are given directly.

Experimental setup. We evaluate recurrent TinyLlama, recurrent OLMo (McLeish et al., 2025), Ouro-1.4B, and Ouro-2.6B (Zhu et al., 2025), without further training. We report $\{1, 2, 4, 8\}$ loops for TinyLlama/OLMo and 1–4 loops for Ouro, as further increasing loops yields limited improvements (McLeish et al., 2025; Zhu et al., 2025).

We measure *multi-step implicit reasoning capability* using arithmetic expressions such as $((3+4)\times 2)-5$. The model needs to generate the final answer without writing intermediate steps. Concretely,

we use expressions that require two to four calculations, with all intermediate and final results in 0–99. We use 3-shot prompting, with each example showing only the final answer. We measure *single-step reasoning capability within a CoT* using GSM-∞ problems that also require two to four calculations, with all quantities restricted to 0–99. We provide the question and correct preceding steps, then ask the model to complete a single calculation, such as `total = parrots + owls =`. This requires using values from the context to calculate the result. As a control, we ask the model to perform the same calculation with those values inserted directly, such as `3 + 2 =`, without the question or CoT. Both conditions use 1-shot prompting. We include more details in Appendix I.

Results. *Stronger multi-step implicit reasoning improves explicit reasoning.* Figure 6 shows the aggregated results across all models and loop counts. Surprisingly, although every model achieves 100% accuracy at every loop count when given the numbers directly (numeric control), with fewer loops, models still struggle with the same calculations within a CoT. In contrast, with more loops, multi-step implicit reasoning and single-step accuracy within a CoT both improve substantially, with a high correlation (Pearson's $r = 0.846$). Appendix I includes results for individual models, with even higher correlations within each model family ($r = 0.952$ and 0.990). These results suggest that improved implicit reasoning leads to stronger explicit reasoning in models not trained with SeDo.

6 Discussion

Why implicit reasoning helps. It may seem surprising that learning to infer omitted steps helps models that write out every step. We offer two possible explanations. First, predicting further ahead discourages shortcuts. Bachmann & Nagarajan (2024) show that a model can predict the next token nearly perfectly without representing the underlying problem; to mitigate this, previous work trains models to predict more tokens ahead (Gloeckle et al., 2024; Mahajan et al., 2026). However, in CoT reasoning, steps are better units: in Maze, for example, predicting the parenthesis after `close` only teaches the format, while predicting the next search event requires learning the algorithm. SeDo therefore applies this principle to CoT steps: its joint horizon targets train models to predict further steps by first performing the omitted steps implicitly. Consistent with this, token-level deletion and masking work worse than SeDo, especially on GSM-∞, whose steps are longest.

Second, CoT reasoning is not fully explicit. Even when every step is written out, the model still needs to perform implicit state-tracking from context. For example, a calculation is harder within a GSM-∞ CoT than in isolation, because its values must be found in earlier steps (§5.3). In SeDo, this capability is deliberately trained: the models need to correctly infer the current state even when some steps are omitted. This is consistent with SeDo's especially large gains on Maze from complete prefixes: since the search state, e.g., the nodes visited and the queue, is never written out, the model must always infer it implicitly to produce the solution path.

Extending SeDo to broader tasks. Applying SeDo to broader tasks raises two questions. First, which steps should be omitted? The model should be able to infer the missing step from the earlier context. For example, it cannot recover information from a file it has not seen. Non-unique steps are also harder to predict: in Maze, SeDo needs more data when the traces are randomized. The omitted steps should also be small enough to learn: our DEPO experiments show that learning many steps without intermediate supervision is difficult. But they should not be too small either, as removing individual tokens is less effective than removing steps. Second, how can we obtain enough training data with these steps? Models need many examples of similar steps to learn to perform them internally. Our results with self-generated CoTs suggest one promising direction: first train a model on the available complete traces, then use it to generate more traces of internalizable steps.

7 Conclusions

We have shown that learning to reason more implicitly improves explicit reasoning on harder problems. Our method, SeDo, trains models on both complete CoTs and partial CoTs with randomly omitted steps. SeDo makes implicit reasoning easier to learn and improves CoT reasoning on problems that require more steps. We have also shown that stronger implicit reasoning improves the accuracy of individual CoT steps even without SeDo training. Learning from less data and extending SeDo to broader tasks where steps are less clearly defined are promising future directions.

AI USE STATEMENT

In this work, we used generative AI tools to help refine research hypotheses, provide feedback on experimental design, and assist with the implementation, including data generation pipeline and training pipeline. We did not use generative AI tools to formulate mathematical claims, provide critical ingredients for mathematical proofs, assist in writing proofs, or interpret experimental results. Other required-disclosure tasks, including synthetic dataset generation, theoretical model or conceptual framework development, translation, dataset cleaning or reformatting, and qualitative or thematic data analysis, were not applicable to this work.

Additionally, we used generative AI tools to create or modify scientific figures, polish the writing, and suggest improvements to the structure of the paper. We reviewed all AI-assisted work. In particular, we manually reviewed AI-assisted code used in the core implementation and verified its correctness through testing and empirical checks.

We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide detailed information to facilitate reproduction of our results. Appendix A explains SeDo and SeDo-R with implementation details. Appendices B, C, and D describe data generation, training settings, and evaluation. Appendices H and I describe the details of the probing and pretrained looped Transformer experiments.

REFERENCES

Zeyuan Allen-Zhu. Physics of language models: Part 4.1, architecture design and the magic of Canon layers. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 42349–42369. Curran Associates, Inc., 2025. doi: 10.52202/085713-1410. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/3c4141c12660ad3625eb4ae845e0a6f9-Paper-Conference.pdf.

Awni Altabaa, Omar Montasser, and John Lafferty. CoT information: Improved sample complexity under Chain-of-Thought supervision. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://arxiv.org/abs/2505.15927.

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. Exploring length generalization in large language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/fb7451e43f9c1c35b774bcfad7a5714b-Abstract-Conference.html.

Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 2296–2318, 2024. URL https://proceedings.mlr.press/v235/bachmann24a.html.

Changyu Chen, Xiting Wang, Ting-En Lin, Ang Lv, Yuchuan Wu, Xin Gao, Ji-Rong Wen, Rui Yan, and Yongbin Li. Masked thought: Simply masking partial reasoning steps can improve mathematical reasoning learning of language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5872–5900, 2024. doi: 10.18653/v1/2024.acl-long.320. URL https://aclanthology.org/2024.acl-long.320/.

Graham Cooper and John Sweller. Effects of schema acquisition and rule automation on mathematical problem-solving transfer. *Journal of Educational Psychology*, 79(4):347–362, 1987. doi: 10.1037/0022-0663.79.4.347.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*, 2023. URL https://arxiv.org/abs/2311.01460.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit CoT to implicit CoT: Learning to internalize CoT step by step. *arXiv preprint arXiv:2405.14838*, 2024. URL https://arxiv.org/abs/2405.14838.

Yupei Du, Philipp Mondorf, Silvia Casola, Yuekun Yao, Robert Litschko, and Barbara Plank. Reason to rote: Rethinking memorization in reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 8659–8679, 2025. doi: 10.18653/v1/2025.emnlp-main.437. URL https://aclanthology.org/2025.emnlp-main.437/.

Gemma Team. Gemma 3 Technical Report. *arXiv preprint arXiv:2503.19786*, 2025. URL https://arxiv.org/abs/2503.19786.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 15706–15734, 2024. URL https://proceedings.mlr.press/v235/gloeckle24a.html.

Steve Hanneke, Idan Mehalel, and Shay Moran. Sample complexity of autoregressive reasoning: Chain-of-thought vs. end-to-end. *arXiv preprint arXiv:2604.12013*, 2026. URL https://arxiv.org/abs/2604.12013.

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. In *Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2412.06769.

Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Cheng Jiayang, Yue Zhang, Xipeng Qiu, and Zheng Zhang. Can language models learn to skip steps? In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/504fa7e518da9d1b53a233ed20a38b46-Abstract-Conference.html.

Divyat Mahajan, Sachin Goyal, Badr Youbi Idrissi, Mohammad Pezeshki, Ioannis Mitliagkas, David Lopez-Paz, and Kartik Ahuja. Beyond multi-token prediction: Pretraining LLMs with future summaries. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.14751.

Alan Malek, Jiawei Ge, Nevena Lazic, Chi Jin, András György, and Csaba Szepesvári. Frontier LLMs still struggle with simple reasoning tasks. *arXiv preprint arXiv:2507.07313*, 2025. URL https://arxiv.org/abs/2507.07313.

Sean McLeish, Ang Li, John Kirchenbauer, Dayal Singh Kalra, Brian R. Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Jonas Geiping, Tom Goldstein, and Micah Goldblum. Teaching pretrained language models to think deeper with retrofitted recurrence. *arXiv preprint arXiv:2511.07384*, 2025. URL https://arxiv.org/abs/2511.07384.

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. Arithmetic without algorithms: Language models solve math with a bag of heuristics. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.21272.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021. URL https://arxiv.org/abs/2112.00114.

Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. Fine-tuning enhances existing mechanisms: A case study on entity tracking. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2402.14811.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. CODI: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, 2025. doi: 10.18653/v1/2025.emnlp-main.36. URL https://aclanthology.org/2025.emnlp-main.36/.

Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3603. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/9b26ad15462c81548c0689188d2e8018-Abstract-Conference.html.

DiJia Su, Sainbayar Sukhbaatar, Michael Rabbat, Yuandong Tian, and Qinqing Zheng. Dualformer: Controllable fast and slow thinking by learning with randomized reasoning traces. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/ed45d6a03de84cc650cae0655f699356-Abstract-Conference.html.

Niels A. Taatgen. The nature and transfer of cognitive skills. *Psychological Review*, 120(3):439–471, 2013. doi: 10.1037/a0033138.

Petar Veličković, Adrià Puigdomènech Badia, David Budden, Razvan Pascanu, Andrea Banino, Misha Dashevskiy, Raia Hadsell, and Charles Blundell. The CLRS algorithmic reasoning benchmark. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 22084–22102. PMLR, 2022. URL https://proceedings.mlr.press/v162/velickovic22a.html.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, 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, pp. 24824–24837, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. SIM-CoT: Supervised implicit chain-of-thought. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/5d087955ee13fe9a7402eedec879b9c3-Abstract-Conference.html.

Kaiyue Wen, Huaqing Zhang, Hongzhou Lin, and Jingzhao Zhang. From sparse dependence to sparse attention: Unveiling how Chain-of-Thought enhances transformer sample efficiency. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/fa6d4d2020aac4bd8f7cdb2771fc1ae2-Abstract-Conference.html.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a. URL https://arxiv.org/abs/2505.09388.

Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. Do large language models latently perform multi-hop reasoning? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10210–10229, 2024. doi: 10.18653/v1/2024.acl-long.550. URL https://aclanthology.org/2024.acl-long.550/.

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, pp. 3971–3992, 2025b. doi: 10.18653/v1/2025.findings-acl.205. URL https://aclanthology.org/2025.findings-acl.205/.

Yuekun Yao, Yupei Du, Dawei Zhu, Michael Hahn, and Alexander Koller. Language models can learn implicit multi-hop reasoning, but only if they have lots of training data. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 9684–9702, 2025. doi: 10.18653/v1/2025.emnlp-main.490. URL https://aclanthology.org/2025.emnlp-main.490/.

Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Beyond autoregression: Discrete diffusion for complex reasoning and planning. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.14157.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*, 2024. URL https://arxiv.org/abs/2407.06023.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/537d5aa768c2d534016a4d06f87bc8fb-Abstract-Conference.html.

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Josh Susskind, Samy Bengio, and Preetum Nakkiran. What algorithms can transformers learn? a study in length generalization. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/45ed1a72597594c097152ef9cc187762-Abstract-Conference.html.

Yang Zhou, Hongyi Liu, Zhuoming Chen, Yuandong Tian, and Beidi Chen. GSM-∞: How do your LLMs behave over infinitely increasing reasoning complexity and context length? In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 78933–78983. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhou25m.html.

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. URL https://arxiv.org/abs/2510.25741.

A SEDO AND SEDO-R

Steps. We define a step as a unit of reasoning from the task's CoT: an intermediate node in DEPO, a search event in Maze, a partition decision or state update in Quickselect, or a line of calculation in GSM-∞. To build a partial CoT, we select each step independently with probability p, *without* selecting the question or the final answer (in Maze, we also keep the first and last search events).

SeDo and SeDo-R. SeDo simply deletes the selected steps and trains with standard next-token prediction on the shorter sequence. In contrast, SeDo-R replaces each selected step with one mask token and uses the masked sequence as context for predicting later steps. Moreover, it also trains the model to predict each selected step from the context just before its mask, in which earlier selected steps are still masked; we will describe the efficient implementation below.

Example. Assume we have three steps of CoTs:

1. `A = 3 + 2 = 5`,
2. `B = A * 2 = 10`,
3. `C = B - 1 = 9`.

And assume we randomly selected B to omit (in SeDo) or mask (in SeDo-R): SeDo trains on the sequence of `A = 3 + 2 = 5` and `C = B - 1 = 9`, using an ordinary next-token prediction loss. However, for SeDo-R, there are two modes: first, when predicting the masked steps, i.e., the line of `B = A * 2 = 10`, the model observes `A = 3 + 2 = 5`; however, when predicting a later step `C = B - 1 = 9`, the model observes `A = 3 + 2 = 5` and `<MASK>`, where the second step is replaced by a special mask token.

Implementation. We compute all SeDo-R predictions in one forward pass by appending each selected step as a separate branch and using an attention mask so that each branch sees only the context before its mask, with positions continuing from that point. At test time, both methods use ordinary autoregressive generation.

B DEPO EXPERIMENTAL DETAILS

Data and evaluation. Following Allen-Zhu (2025), we generate training examples online with a maximum of 225 nodes, hop counts up to 16, and ten queries per example. Each node name contains one or two tokens drawn from a vocabulary of 50 symbols. We use a CoT that lists intermediate nodes before the final answer. Table 1 reports results from the final 150,000-update checkpoint.

Models and optimization. Following Allen-Zhu (2025), Canon-Llama has eight layers, a hidden size of 512, eight attention heads, and the ABCD Canon configuration. GPT2-RoPE has twelve layers, a hidden size of 768, twelve attention heads, and an intermediate size of 3,072. It uses rotary position embeddings, RMSNorm, and an ungated SiLU feed-forward network.

Both architectures are initialized from scratch. We use 2,048-token training blocks and a global batch of 128 blocks. Training uses AdamW with weight decay 0.03, $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-6}$, gradient clipping at 1.0, and bfloat16 precision. Except for iCoT-SI, which specifies a curriculum, we warm up for 1,000 updates and then use cosine learning-rate decay to 0.1 times the peak rate. We search learning rates over $\{3 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}, 2 \times 10^{-3}\}$.

Shared schedule. Answer-only training uses no intermediate nodes. All other methods except iCoT-SI start from a mixture of answer-only examples and CoTs, then gradually replace the CoTs with answer-only examples using a cosine schedule from update 30,000 to 90,000. After update 90,000, all examples are answer-only.

CoT mixtures. Full-fade starts with 20% answer-only examples and 80% complete CoTs. SeDo starts with 20% answer-only examples, 20% complete CoTs, and 60% partial CoTs, in which each intermediate node is omitted independently with probability 0.5. SeDo-R uses the same mixture as SeDo.

Multi-token objectives. MTP and FSP use the same mixture as Full-fade. MTP predicts each of the next four tokens with a separate one-layer head. FSP adds one one-layer head that predicts which tokens appear in the next four positions, trained with binary cross-entropy and a loss weight of 1. In total, MTP adds three layers and FSP adds one, while next-token prediction still uses eight layers for Canon-Llama and twelve for GPT2-RoPE.

iCoT-SI. iCoT-SI instead removes tokens from the beginning of the CoT over training. We search over removing one token each {1000, 2000, 3000, 4000} steps. We use its smoothing parameter of 4, keep the learning rate constant, and reset the optimizer state whenever the number of removed tokens increases.

C RL AND SELF-GENERATED COT DETAILS

Setup. Both experiments start from Qwen3-0.6B checkpoints after 50,000 SFT updates, trained on 100,000 Quickselect problems or 25,000 mazes. Each task uses a new pool of 32,000 problems, separate from the evaluation sets.

RL training. We run GRPO separately from the Full and SeDo checkpoints, without omitting or masking steps. Each run has 2,000 updates, each with 16 problems and eight completions per problem (128 completions per update). We use a learning rate of 2×10^{-6} with 50 warmup updates and a constant rate afterward, AdamW with weight decay 0.03, $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-6}$, gradient clipping at 1.0, no KL penalty, and bfloat16 precision.

RL rollouts and rewards. Rollouts use temperature 1, top-p 1, and an 8,192-token context, with at most 7,168 completion tokens for Quickselect and 6,144 for Maze. In Quickselect, the reward is 1 if both the selected value and the terminal array are correct, and 0 otherwise. In Maze, a valid final path earns 1, plus 0.5 if it is also shortest.

Self-generated data. The Full SFT checkpoint solves each problem in the pool with temperature 1 and top-p 1, using up to eight attempts and keeping the first solution that passes the checks. In Quickselect, the answer and terminal array must be correct, and the trace must be parsable with at least one partition operation. In Maze, the trace must be parsable and end in a shortest path. Problems without an accepted solution are discarded, and we train Identity and SeDo on the same set of accepted solutions.

Continued training. Both conditions start from the Full SFT checkpoint and train on copies of the accepted solutions. Identity uses only complete copies. SeDo keeps about half of the copies complete, and in the rest omits each step independently with probability 0.3. We train for 20,000 updates with a batch of four packed sequences of up to 8,192 tokens each, using a learning rate of 5×10^{-5}, 400 warmup updates, and cosine decay.

D EXPERIMENTAL DETAILS FOR COMPLEXITY GENERALIZATION

Data. *Maze.* We use 25,000 25×25 mazes with $75 \leq H < 200$. CoTs come from deterministic A* search: each records the search events and ends with the solution path. *Quickselect.* We use 100,000 problems with $60 \leq H < 140$. CoTs come from stable Quickselect, using the last element of the active array as the pivot. For each partition, the CoT records whether each other element goes left or right, and then the updated array and active range. The final line gives the selected value and the terminal array. *GSM-∞.* We use 387,500 problems with $16 \leq H < 26$, generated with the official forward generator and its three templates (Crazy Zootopia, Teachers in School, and Movie Festival Awards), with irrelevant statements removed. CoTs follow the computational graph in topological order, with each line giving a variable name, its expression, and its value. The final line gives the answer after an `[ANSWER]` marker.

We call each training copy of a problem a view. SeDo and SeDo-R create partial view of a sample, by omitting or masking each step independently with probability 0.3, as described in Appendix A. In Maze and Quickselect, half of the views are partial. In GSM-∞, each problem has four views: all

complete for Full, and one complete and three partial for SeDo and SeDo-R. Token Drop randomly removes individual tokens, matching the number removed by SeDo. MFT masks individual tokens, also with probability 0.3.

Optimization. We train with AdamW in bfloat16 precision and compute loss only on completion tokens. We pack examples into sequences using the training settings in Table 5. The learning rate increases linearly during warmup, then follows cosine decay to zero for Maze and Quickselect and to 10% of its peak for GSM-∞. We choose the peak learning rate from $\{1 \times 10^{-5}, 3 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}\}$ using validation performance. Table 6 lists the chosen rates. For each task and model, all five methods use the same rate.

Task	Updates	Global batch	Pack length	Warmup
Maze	50,000	4	8,192	1,000
Quickselect	50,000	4	8,192	400
GSM-∞	20,000	32	2,048	500

Table 5: Training settings. Global batch size counts packed sequences per update.

Task	Qwen3-0.6B	Qwen3-1.7B	Gemma3-1B
Maze	5×10^{-5}	5×10^{-5}	5×10^{-5}
Quickselect	5×10^{-5}	5×10^{-5}	5×10^{-5}
GSM-∞	5×10^{-5}	3×10^{-5}	3×10^{-5}

Table 6: Selected peak learning rates.

Evaluation. We evaluate on held-out test problems with greedy decoding (p@1) and sampling. For sampling, we use temperature 1 and top-$p = 1$ on Maze and Quickselect, and temperature 0.6, top-$p = 0.95$, and top-$k = 20$ on GSM-∞. We evaluate path validity in Maze and final-answer correctness in Quickselect and GSM-∞. We report p@k, the estimated probability that at least one of k sampled solutions is correct, averaged over test problems.

E GENERALIZATION RESULTS ACROSS MODELS

Tables 7–9 report results for all three models. For each task, the first horizon range is in-distribution and the remaining two are out-of-distribution.

	Maze								
H	$100 \leq H < 200$			$200 \leq H < 525$			$525 \leq H < 650$		
Method	p@1	p@8	p@16	p@1	p@8	p@16	p@1	p@8	p@16
Qwen3-1.7B									
Full	96.49	97.93	98.55	50.00	62.66	67.19	6.00	12.33	15.83
Token Drop	98.14	99.59	100.00	58.05	74.69	80.78	7.00	16.17	21.83
MFT	96.07	97.93	97.93	52.97	66.56	70.94	9.00	19.17	23.00
SeDo	97.52	99.59	100.00	**61.33**	**80.00**	**85.70**	11.83	**28.50**	**38.83**
SeDo-R	95.66	98.14	99.17	58.75	74.69	77.66	**12.67**	28.17	34.00
Qwen3-0.6B									
Full	95.04	97.31	97.93	46.64	61.56	65.70	4.00	12.83	17.00
Token Drop	97.73	99.59	99.59	47.81	67.50	74.61	6.50	15.17	20.33
MFT	96.69	98.97	99.17	53.83	70.47	74.69	7.83	18.83	23.17
SeDo	96.07	99.79	100.00	**57.66**	**78.67**	**84.92**	12.67	27.50	**36.00**
SeDo-R	96.49	98.55	98.76	57.50	73.52	76.95	**14.17**	**29.17**	33.33
Gemma3-1B									
Full	89.26	94.63	95.25	38.36	56.09	60.86	5.00	14.33	17.00
Token Drop	95.04	98.55	99.17	43.75	67.19	73.59	6.00	14.33	19.00
MFT	92.77	96.90	97.52	43.75	59.38	62.66	6.67	15.00	18.50
SeDo	93.39	98.76	99.17	44.14	**68.83**	**75.86**	7.83	22.33	**30.00**
SeDo-R	95.25	97.52	98.14	**52.50**	66.02	70.55	**11.83**	**23.33**	27.33

Table 7: Maze test performance (%) across ID and two OOD complexity ranges. Bold marks the best result within each model and OOD column.

	Quickselect								
H	$60 \leq H < 140$			$140 \leq H < 300$			$300 \leq H < 380$		
Method	p@1	p@8	p@16	p@1	p@8	p@16	p@1	p@8	p@16
Qwen3-1.7B									
Full	96.00	99.50	99.75	50.38	74.88	81.75	24.50	58.75	66.75
Token Drop	98.50	100.00	100.00	60.75	82.25	90.25	33.75	66.25	80.25
MFT	97.75	100.00	100.00	63.38	79.75	82.88	34.75	60.25	68.75
SeDo	98.50	99.75	100.00	68.75	**94.12**	**97.12**	41.50	**88.25**	**96.25**
SeDo-R	98.75	100.00	100.00	**70.00**	86.75	89.88	**43.75**	70.50	76.00
Qwen3-0.6B									
Full	80.75	99.75	100.00	33.50	61.75	68.88	14.25	34.00	39.50
Token Drop	97.50	99.50	99.75	66.50	85.38	93.00	41.75	75.75	86.75
MFT	98.75	99.75	99.75	62.62	80.75	85.12	31.25	57.25	65.50
SeDo	99.25	99.75	100.00	**72.12**	**94.88**	**97.38**	**54.25**	**86.00**	**95.00**
SeDo-R	98.50	99.75	100.00	70.62	83.88	86.62	44.75	68.00	73.25
Gemma3-1B									
Full	91.00	98.50	99.25	39.25	64.38	72.75	16.00	42.25	52.75
Token Drop	99.25	99.50	100.00	66.62	89.88	95.62	35.75	74.00	86.00
MFT	98.75	100.00	100.00	**75.12**	86.62	89.50	**53.50**	73.75	80.25
SeDo	97.00	99.75	99.75	66.25	**91.88**	**96.88**	41.50	**81.00**	**92.25**
SeDo-R	98.00	99.00	99.00	67.62	82.25	85.12	36.50	60.50	66.50

Table 8: Quickselect test performance (%) across ID and two OOD complexity ranges. Bold marks the best result within each model and OOD column.

	GSM-∞								
H	$16 \leq H < 26$			$26 \leq H < 31$			$31 \leq H < 36$		
Method	p@1	p@8	p@16	p@1	p@8	p@16	p@1	p@8	p@16
Qwen3-1.7B									
Full	92.50	95.17	96.04	55.42	70.04	75.00	40.00	55.90	60.83
Token Drop	89.38	93.88	95.00	49.58	69.55	76.25	34.17	56.36	62.08
MFT	91.88	94.82	95.83	50.42	69.19	75.00	39.17	62.57	69.58
SeDo	93.54	97.01	98.12	**60.83**	**79.98**	**84.58**	**49.17**	**70.08**	**77.08**
SeDo-R	93.54	96.97	97.71	58.75	77.40	82.08	47.08	68.18	73.33
Qwen3-0.6B									
Full	90.83	94.08	95.00	50.00	68.64	73.33	37.50	56.15	62.08
Token Drop	92.08	95.13	96.04	59.58	71.59	75.42	42.08	63.77	69.17
MFT	91.88	94.88	95.83	55.42	71.17	75.00	45.42	61.91	67.50
SeDo	94.38	97.15	97.50	**69.17**	80.92	**87.08**	55.42	**74.21**	**79.58**
SeDo-R	93.96	96.20	96.67	**69.17**	**81.88**	85.83	**57.92**	73.06	77.08
Gemma3-1B									
Full	89.58	92.19	92.92	35.00	51.10	55.83	18.75	33.94	39.58
Token Drop	83.33	90.51	92.08	26.25	42.28	47.92	12.92	26.03	32.50
MFT	83.12	90.84	92.71	35.00	49.66	55.42	16.25	31.80	40.42
SeDo	90.62	95.04	96.46	**44.17**	**62.88**	**67.92**	**30.42**	46.59	**54.58**
SeDo-R	90.83	95.03	96.25	43.33	59.01	64.17	29.58	**47.19**	54.17

Table 9: GSM-∞ test performance (%) across ID and two OOD complexity ranges. Bold marks the best result within each model and OOD column.

F LEARNING IMPLICIT REASONING

F.1 DEPO LEARNING PROGRESS

Figure 7 shows how many hops Canon-Llama learns to reason over during training on DEPO. SeDo reaches 80% accuracy across the tested hop counts up to 16 after 75k updates, compared with 110k for MTP, 145k for iCoT-SI, and 150k for FSP.

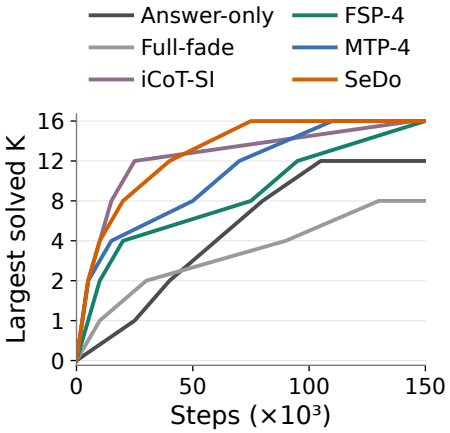
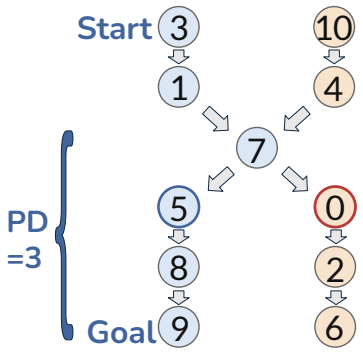


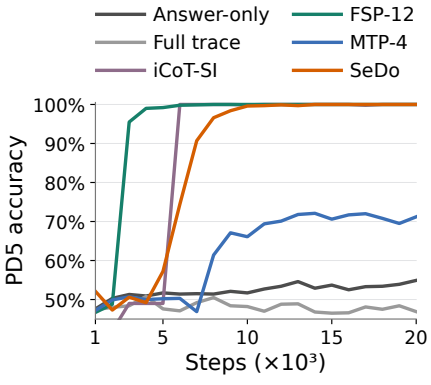
Figure 7: DEPO learning progress. The y axis shows the largest tested hop count solved (i.e., with $> 80\%$ accuracy and does not fall below afterwards). FSP and MTP add one and three auxiliary layers, respectively.

F.2 IMPLICIT LOOKAHEAD

We also test whether joint prediction helps models learn to look ahead implicitly, using the graph task from Ye et al. (2025). Each problem gives the shuffled edges of two six-node directed paths that intersect at one node, together with a start and a goal. The model must choose the path that reaches the goal and output its edges in order. Planning Distance (PD) is the distance from the choice to the goal. At $\mathrm{PD} = 5$, choosing the first edge requires looking five edges ahead. We train a six-layer GPT-2-style model from scratch with depth-first-search traces and evaluate exact-match accuracy without generating a CoT.



(a) Lookahead task.



(b) Learning implicit lookahead.

Figure 8: Implicit lookahead. Left: an instance with $\mathrm{PD} = 3$; blue marks the solution path and orange the distractor. Right: direct-answer validation accuracy at $\mathrm{PD} = 5$. Random guessing achieves 50%.

Answer-only and full-CoT training remain near 50%. MTP-4 reaches 71.2% at 20k updates, while SeDo, FSP-12, and iCoT-SI reach 100%. These results show that joint prediction also helps models learn to look ahead implicitly, suggesting its capability in planning.

G PREDICTION FROM COT PREFIXES ACROSS PROBLEM COMPLEXITIES

The tables below show how accurately each model continues a complete or incomplete CoT on easier (ID) and harder (OOD) problems. We evaluate the solution path in Maze, the next partition state in Quickselect, and the next calculation in GSM-∞.

Task	H	Incomplete			Complete		
		Full	SeDo	Gain (%)	Full	SeDo	Gain (%)
Maze	< 200	15.19	**96.35**	+534.2	95.58	**96.73**	+1.2
	$[200, 525)$	4.30	**56.41**	+1212.7	48.44	**59.53**	+22.9
	$[525, 650)$	0.33	**17.67**	+5200.0	5.50	**13.83**	+151.5
Quickselect	< 140	5.26	**82.04**	+1458.8	100.00	100.00	0.0
	$[140, 300)$	3.38	**64.25**	+1803.7	82.75	**89.00**	+7.6
	$[300, 380)$	3.00	**49.00**	+1533.3	71.25	**79.50**	+11.6
GSM-∞	$[16, 26)$	62.71	**93.96**	+49.8	97.50	**98.96**	+1.5
	$[26, 31)$	53.75	**78.75**	+46.5	93.75	**96.25**	+2.7
	$[31, 36)$	52.92	**76.25**	+44.1	90.42	**94.17**	+4.1

Table 10: Qwen3-0.6B greedy accuracy (%) from complete and incomplete CoT prefixes. The first range for each task is ID; the other two are OOD. Gain is the relative improvement of SeDo over Full.

Task	H	Incomplete			Complete		
		Full	SeDo	Gain (%)	Full	SeDo	Gain (%)
Maze	< 200	15.38	**95.38**	+520.0	96.92	**97.88**	+1.0
	$[200, 525)$	2.58	**60.94**	+2263.6	51.02	**62.19**	+21.9
	$[525, 650)$	0.33	**16.17**	+4750.0	6.50	**13.17**	+102.6
Quickselect	< 140	4.33	**84.52**	+1850.0	100.00	100.00	0.0
	$[140, 300)$	3.75	**63.63**	+1596.7	85.88	**90.00**	+4.8
	$[300, 380)$	3.00	**49.75**	+1558.3	77.00	**78.50**	+1.9
GSM-∞	$[16, 26)$	62.71	**91.46**	+45.8	98.54	**98.96**	+0.4
	$[26, 31)$	50.42	**78.75**	+56.2	93.33	**94.17**	+0.9
	$[31, 36)$	51.25	**77.08**	+50.4	90.83	**93.75**	+3.2

Table 11: Qwen3-1.7B greedy accuracy (%) from complete and incomplete CoT prefixes. The first range for each task is ID; the other two are OOD. Gain is the relative improvement of SeDo over Full.

Task	H	Incomplete			Complete		
		Full	SeDo	Gain (%)	Full	SeDo	Gain (%)
Maze	< 200	21.73	**90.96**	+318.6	91.15	**94.62**	+3.8
	$[200, 525)$	5.08	**47.97**	+844.6	41.41	**47.97**	+15.8
	$[525, 650)$	0.67	**10.67**	+1500.0	5.83	**11.17**	+91.4
Quickselect	< 140	4.02	**85.76**	+2030.8	99.07	**99.38**	+0.3
	$[140, 300)$	3.38	**59.88**	+1674.1	84.75	**88.00**	+3.8
	$[300, 380)$	3.25	**52.00**	+1500.0	74.25	**77.50**	+4.4
GSM-∞	$[16, 26)$	62.71	**89.38**	+42.5	97.08	**97.71**	+0.6
	$[26, 31)$	48.75	**71.67**	+47.0	86.25	**89.58**	+3.9
	$[31, 36)$	47.08	**70.00**	+48.7	81.25	**88.75**	+9.2

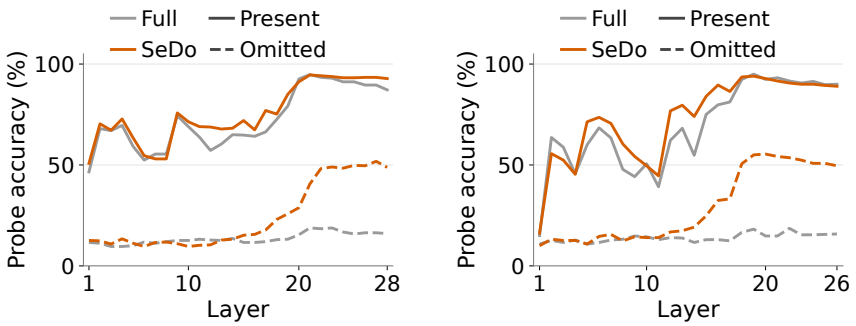
Table 12: Gemma3-1B greedy accuracy (%) from complete and incomplete CoT prefixes. The first range for each task is ID; the other two are OOD. Gain is the relative improvement of SeDo over Full.

H INTERMEDIATE-VALUE PROBING

Data construction. We generate new GSM-∞ problems within the training range ($16 \leq H < 26$). For each problem, we select a variable V used later in the CoT and change the input numbers so that V takes values from 10 to 29. The calculation structure and question wording stay the same. For each variant, we create two CoTs: one with V's calculation and one without it. Before the probing position, V's value appears only in its defining calculation, so in the CoT without it, the value does not appear at all. We use 20k training instances and 500 each for validation and test, balanced across target values.

Probe training. We keep the Full and SeDo model weights fixed. We find the first later calculation that uses V and extract the hidden state just before its result: for example, at the second equals sign in `U = V * 2 =`. We train a separate linear probe for each layer, model, and condition (with or without V's calculation) to predict the value of V. We standardize each hidden-state dimension using its mean and standard deviation on the probe's training data. We select probe hyperparameters on validation data using cross-entropy and report accuracy on the held-out test set.

Additional models. Figure 9 shows the same probing experiment for Qwen3-0.6B and Gemma3-1B. For both models, probes predict omitted values more accurately from SeDo's hidden states than from Full's, while accuracy is similar when the calculation is present.



(a) Qwen3-0.6B. (b) Gemma3-1B.

Figure 9: Probe accuracy for predicting V just before the result of a calculation that uses it. Solid lines show results with V's calculation present; dashed lines show results with it omitted.

I LOOPED TRANSFORMER RESULTS

I.1 EXPERIMENTAL SETUP

Models. We use pretrained recurrent TinyLlama and OLMo (McLeish et al., 2025) at $R \in \{1, 2, 4, 8\}$ loops, and Ouro-1.4B and Ouro-2.6B (Zhu et al., 2025) at $R \in \{1, 2, 3, 4\}$. Each configuration uses a fixed number of loops, with adaptive exit disabled for Ouro. For TinyLlama and OLMo, we average accuracy over three runs with different random initial hidden states; Ouro uses one run.

Answer-only arithmetic. We give the model an arithmetic expression and ask it to return the answer without writing out the calculations. For example, $((3{+}4){\times}2){-}5$ requires three calculations, but the model should output just `9`. This tests how many calculations it can perform internally. We call the number of calculations the expression's *depth*. We generate 300 expressions for each depth from 1 to 8. Half use addition and subtraction, and half also include multiplication. The calculations form a chain: in the example above, we first get 7, then multiply it by 2 to get 14, and finally subtract 5 to get 9. All intermediate and final answers are between 0 and 99. Each prompt includes three examples showing an expression and its answer (3-shot).

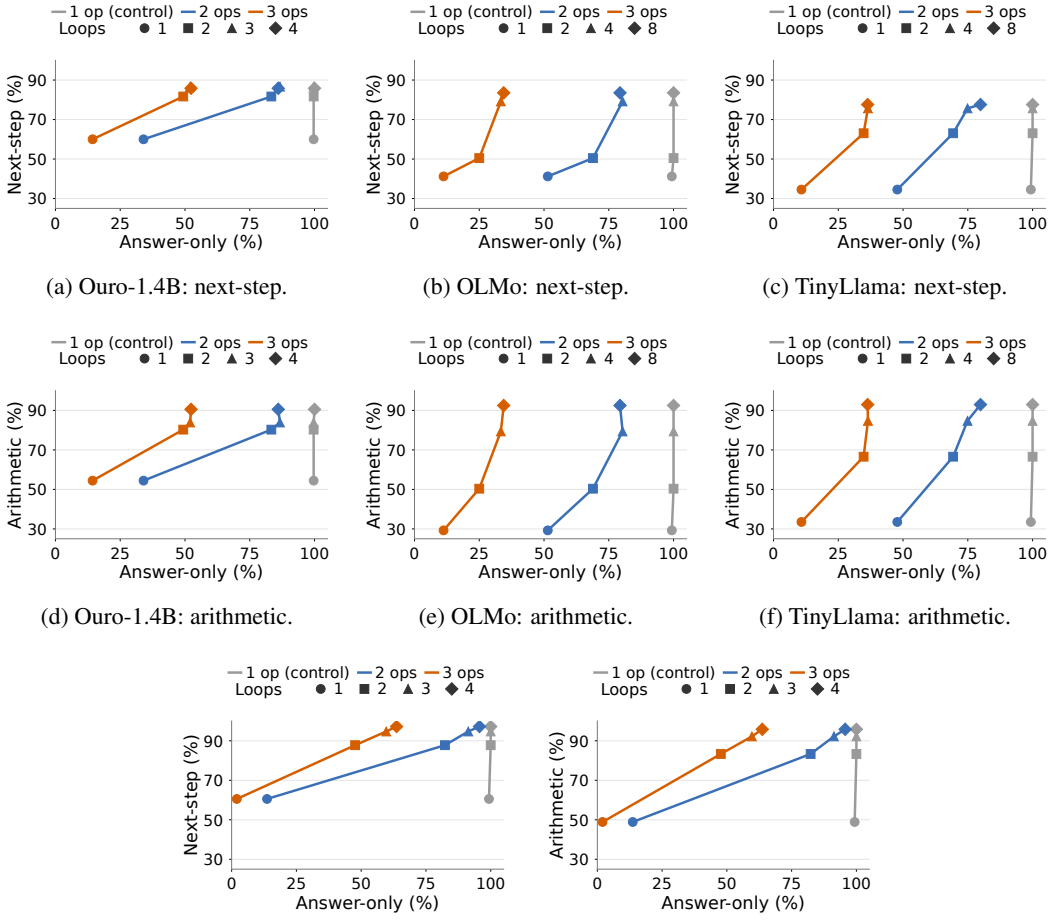
CoT single-step reasoning. We test whether the model can correctly perform one calculation within a CoT. We provide the question and all correct steps before that calculation. For example, suppose these steps establish that there are three parrots and two owls, and the next step adds them together. We test three versions:

- *Step-value:* give `total = parrots + owls =` and ask the model to complete it with `5`.
- *Next-step:* give `total =` and ask the model to complete the calculation, e.g., `parrots + owls = 5`.
- *Numeric control:* give `3 + 2 =` as a standalone question and ask for `5`.

We use 360 GSM-∞ problems requiring 2–4 calculations, with all numbers between 0 and 99. For each number of calculations, 40 problems come from each of the three templates. We test one calculation per problem that uses a result from an earlier step. Each prompt uses one exemplar because of the context window limit. All models and loop counts use the same questions and examples.

Inference. We use the official Hugging Face implementations with BF16 and greedy decoding. The generation limit is 16 tokens for integer answers and 64 for next-step generation; generation ends at EOS or the first newline.

I.2 ADDITIONAL RESULTS

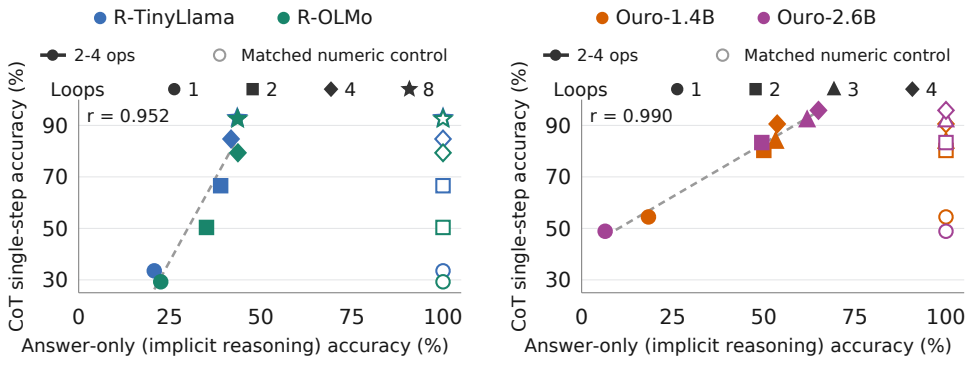


(a) Ouro-1.4B: next-step. (b) OLMo: next-step. (c) TinyLlama: next-step.

(d) Ouro-1.4B: arithmetic. (e) OLMo: arithmetic. (f) TinyLlama: arithmetic.

(g) Ouro-2.6B: next-step. (h) Ouro-2.6B: arithmetic.

Figure 10: Answer-only arithmetic accuracy versus CoT single-step accuracy for each model.



(a) Recurrent TinyLlama and OLMo. (b) Ouro-1.4B and Ouro-2.6B.

Figure 11: Answer-only arithmetic accuracy versus CoT single-step accuracy by model family. Pearson's $r = 0.952$ for recurrent TinyLlama/OLMo and $r = 0.990$ for Ouro.