Enhancing Auto-regressive Chain-of-Thought through Loop-Aligned Reasoning

Qifan Yu[1], Zhenyu He[1], Sijie Li[1], Xun Zhou[2], Jun Zhang[2], Jingjing Xu[2], Di He[1]

[1]State Key Laboratory of General Artificial Intelligence, Peking University, Beijing, China
[2]ByteDance Seed

Correspondence: di_he@pku.edu.cn

Abstract

Chain-of-Thought (CoT) prompting has emerged as a powerful technique for enhancing language model's reasoning capabilities. However, generating long and correct CoT trajectories is challenging. Recent studies have demonstrated that Looped Transformers, a standard Transformer with cross-block parameter-sharing architecture, possess remarkable length generalization capabilities, but their limited generality and adaptability prevent them from serving as an alternative to auto-regressive solutions. To better leverage the strengths of Looped Transformers, we propose **RELAY** (**RE**asoning through **L**oop **A**lignment iterative**Y**). Specifically, we align the steps of Chain-of-Thought (CoT) reasoning with loop iterations and apply intermediate supervision during the training of Looped Transformers. This additional iteration-wise supervision not only preserves the Looped Transformer's ability for length generalization but also enables it to predict CoT reasoning steps for unseen data. Therefore, we leverage this Looped Transformer to generate accurate reasoning chains for complex problems that exceed the training length, which will then be used to fine-tune an auto-regressive model. We conduct extensive experiments, and the results demonstrate the effectiveness of our approach, with significant improvements in the performance of the auto-regressive model. Code will be released at `https://github.com/qifanyu/RELAY`.

1 Introduction

Reasoning plays a central role in shaping effective decision-making processes and guiding problem-solving strategies in artificial intelligence systems. For large language models (LLMs), the most effective way to achieve reasoning is through Chain-of-Thought (Wei et al., 2022; Khot et al., 2022), which generates all intermediate steps token by token until the final answer is reached. However, generating the correct reasoning process using LLMs is challenging. On one hand, the Chain-of-Thought process can be very long, sometimes growing polynomially with respect to the prompt length (Feng et al., 2024; Merrill and Sabharwal, 2024). When the reasoning length exceeds the training data length, it encounters the length generalization problem, where accuracy can drop significantly (Xiao and Liu, 2023; Jin et al., 2024). On the other hand, web data is often noisy, and learning from incorrect trajectories can lead to incorrect answers. While synthetic data could mitigate this issue (Lightman et al., 2024), it requires significant human effort and knowledge to generate and curate.

Recently, an alternative framework has gained attention, known as the looped Transformer (Giannou et al., 2023). In general, the looped Transformer is a standard Transformer model with cross-block parameter sharing, like AlBERT (Lan et al., 2020). In this framework, the input prompt (i.e., the problem) is processed through repeated iterations of the same block, with the number of iterations adaptively determined by the problem complexity. See Figure 1 for an illustration. Several preliminary results (Fan et al., 2024) show that the looped Transformer model has better length generalization capabilities, partially because the increase in problem complexity (e.g., problem length) is not as significant as in the Chain-of-Thought steps.

However, the success of this approach comes with some practical limitations. While determining appropriate loop iterations is feasible for reasoning tasks, it becomes problematic in general tasks, such as translation and summarization. Furthermore, although the looped Transformer can handle specific reasoning tasks, it remains unclear whether it possesses the capability to manage multiple reasoning tasks within a single model. Given these concerns, a natural question arises: if the looped

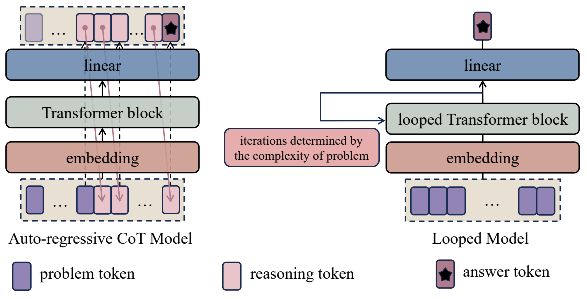


Figure 1: Visualization of Chain-of-Thought (CoT) and looping process. As the complexity of problem increases, in the auto-regressive CoT model, the number of reasoning tokens escalates. In contrast, in the looped model, the number of iterations of the loop block increases.

Transformer is a general reasoner, can we explore ways to integrate its capabilities into the Chain-of-Thought framework of standard auto-regressive models? This integration would allow us to leverage the looped Transformer's strong performance on complex reasoning problems while preserving the versatility that allows auto-regressive models to excel in diverse language tasks.

In this paper, we introduce **RELAY** (**RE**asoning through **L**oop **A**lignment iterative**Y**), a novel framework that leverages looped Transformer's superior capabilities to help auto-regressive models handle longer reasoning chains. At its core, our approach centers on two key innovations. First, we demonstrate empirically that a single looped Transformer model can serve as a general reasoner across multiple tasks while maintaining strong length generalization abilities. Second, we propose an iteration-wise alignment between the looped Transformer and Chain-of-Thought reasoning steps, enabling the looped model to generate accurate reasoning chains for problems beyond training length. These generated reasoning chains can then serve as training data for auto-regressive models, establishing a bridge between the two architectural paradigms.

We conduct extensive experiments demonstrating that our approach significantly improves the reasoning abilities of auto-regressive Transformers through high-quality generated reasoning chains.

2 Related Work

2.1 Auto-regressive LLM with Chain-of-Thought

Chain-of-Thought (CoT) has emerged as a powerful technique for enhancing language models' reasoning capabilities both empirically (Wei et al., 2022; Khot et al., 2022) and theoretically (Feng et al., 2024; Merrill and Sabharwal, 2024), especially in latest models such as OpenAI O1[1], DeepSeek r1 (DeepSeek-AI et al., 2025) and Qwen QwQ[2]. By generating intermediate reasoning steps token by token, these models effectively decompose complex problems into sequential subprocesses. However, two critical challenges persist. First, obtaining high-quality CoT training data remains time-consuming and labor-intensive (Lightman et al., 2024), especially for problems requiring sophisticated reasoning chains. Second, the generation and understanding of extended reasoning sequences can be problematic (Xiao and Liu, 2023; Jin et al., 2024; Mao et al., 2024).

2.2 Looped Transformer

Research on looped Transformers has evolved significantly over recent years. The initial studies by Dehghani et al. (2019) and Lan et al. (2020) demonstrated the effectiveness of parameter sharing across layers in supervised learning and BERT pretraining. This line of research has since expanded in both theoretical and practical directions. On the theoretical front, Giannou et al. (2023) and Xu and Sato (2024) established fundamental properties of looped Transformers, proving their Turing completeness and characterizing their approximation capabilities. Gatmiry et al. (2024) further advanced this understanding by showing how to incorporate inductive biases for learning iterative algorithms, particularly in the context of multi-step gradient descent for in-context learning. Empirically, looped Transformers have shown promising results across various applications. Yang et al. (2024) demonstrated their parameter efficiency in data-fitting tasks, while de Luca and Fountoulakis (2024) and Chen et al. (2024) revealed their potential in graph algorithm simulation and in-context learning enhancement. Notably, Fan et al. (2024) established their superior length generalization capabilities in RASP-L tasks, and Saunshi et al. (2025) further highlighted the power of looped Transformers in complex reasoning tasks by introducing latent thoughts as intermediate representations. In the domain of algorithm learning, Gao et al. (2024) and Yang et al. (2023) explored the effectiveness of leveraging looped Transformers for algorithm representation and learning. While these

[1]https://openai.com/o1
[2]https://qwenlm.github.io/blog/qwq-32b-preview

works have extensively explored various aspects of looped Transformers, our work takes a distinct direction. We specifically focus on leveraging the better length generalization of looped Transformers for helping standard auto-regressive Transformers.

2.3 Approaches for Length Generalization

The capability of Transformers to generalize to longer sequence is influenced by their positional encodings (Press et al., 2022). Recent research has pursued two primary directions to enhance length generalization capabilities of LLMs. The first focuses on developing advanced relative positional encoding schemes (Raffel et al., 2020; Press et al., 2022; Chi et al., 2022; Sun et al., 2023; Chi et al., 2023; Li et al., 2024), while the second explores modifications to positional representations through index granularity adjustments (Chen et al., 2023; Peng et al., 2024) and strategic index shifting (Ruoss et al., 2023; Zhu et al., 2024). These works are orthogonal to the central contributions of this paper. A parallel line of work focuses on improving the reasoning capabilities of LLMs through better training data. These methods typically leverage accessible labels or rewards to generate and filter reasoning steps, selecting those that yield correct solutions or high rewards (Zelikman et al., 2022; Yuan et al., 2024; Singh et al., 2024; Hosseini et al., 2024). However, a critical limitation emerges from LLMs' tendency to generate incorrect or superfluous intermediate reasoning steps while still arriving at correct solutions through chance (Paul et al., 2024). This phenomenon significantly constrains the effectiveness of LLM fine-tuning for complex reasoning tasks (Xia et al., 2024; Zhou et al., 2023).

3 Methodology

3.1 Notation

Any reasoning task can be decomposed into three components: the problem tokens, the reasoning tokens (i.e., chain-of-thought steps), and the answer tokens. Let the problem token sequence be represented as $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$. The Chain-of-Thought (CoT) process generates a sequence of intermediate reasoning tokens $\boldsymbol{z} = [z_1, z_2, \ldots, z_m]$, where n and m denote the number of problem tokens and reasoning tokens respectively. In this work, we focus on a simple setting where the problem's answer is represented by a single token, denoted as y.

CoT Auto-regressive Generation. For auto-regressive generation, the mapping from the problem sequence $\boldsymbol{x}$ to the answer y is performed through generating the intermediate tokens $\boldsymbol{z}$ token by token. Formally, this can be expressed as:

$$z_i \sim P(z_i|\boldsymbol{z}_{<i}, \boldsymbol{x}; \theta), \quad \text{for } i = 1, 2, \ldots, m, \tag{1}$$

where $\boldsymbol{z_{<i}} = [z_1, z_2, \ldots, z_{i-1}]$ represents precedent reasoning tokens. and the final answer is obtained at the final step:

$$y \sim P(y|\boldsymbol{z}, \boldsymbol{x}; \theta). \tag{2}$$

Looped Model. Different from the auto-regressive model that generates explicit tokens to obtain the answer, the looped model implicitly maps the input sequence $\boldsymbol{x}$ to the final answer y by executing the same function (e.g., a multi-layer Transformer block) for T times in the representation space. The number of iterations T depends on the problem comlexity. The forward process consists of three steps: First, the token sequence $\boldsymbol{x}$ is mapped to embeddings through an embedding function h:

$$\boldsymbol{e}_0 = h(\boldsymbol{x}; \theta_{\text{emb}}), \tag{3}$$

where $\boldsymbol{e}_0 \in \mathbb{R}^{d\times n}$ and d is the hidden dimension. Second, the embeddings are iteratively refined through transformation f:

$$\boldsymbol{e}_t = f(\boldsymbol{e}_{t-1}; \theta_{\text{model}}), \quad \text{for } t = 1, 2, \ldots, T, \tag{4}$$

where f is usually a Transformer model and the number of iterations T is adaptively determined based on the problem length. Finally, the answer is predicted through a final-answer prediction head based on the representations in the last layer:

$$y \sim P(y|\boldsymbol{e}_T; \theta_{\text{pred}}). \tag{5}$$

This design enables the model to perform implicit reasoning through iterative refinement in the representation space, where each iteration can automatically capture different aspects of the reasoning process.

As a comparison, the CoT auto-regressive generation derives the final output y by first generating a sequence of intermediate reasoning tokens $\boldsymbol{z} = [z_1, z_2, \ldots, z_m]$ in an auto-regressive manner, where the length m can grow polynomially with input length n (i.e., $m \sim \text{poly}(n)$). This variable and potentially large m poses challenges for positional

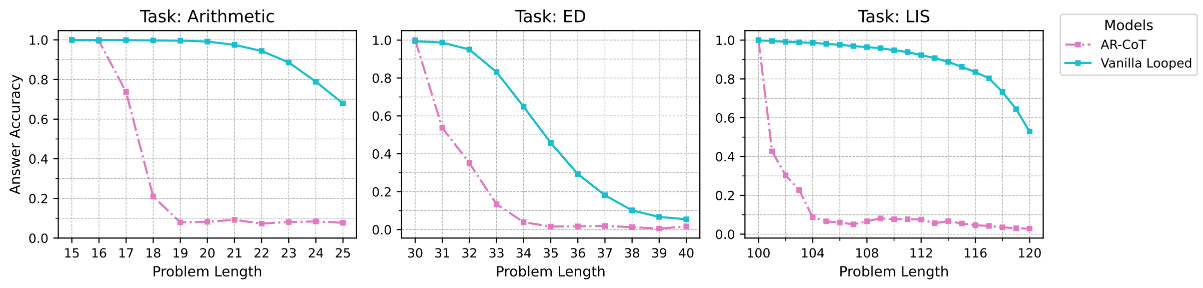


Figure 2: Length generalization performance of looped Transformer versus auto-regressive CoT model on Arithmetic (train: ≤ 15, test: $[15, 25]$), Edit Distance (train: ≤ 30, test: $[30, 40]$), and Longest Increasing Subsequence (train: ≤ 100, test: $[100, 120]$). While both perform well within the training range, the auto-regressive CoT model drops rapidly as problem length increases, whereas the looped model sustains substantially higher accuracy on out-of-distribution long inputs, indicating stronger extrapolation.

encoding to correctly reflect attention relationships, leading to low accuracy for long sequence reasoning. In contrast, the looped model takes a fundamentally different approach. It directly processes the input sequence $\boldsymbol{x}$ and produces output y. The network only needs to handle $\boldsymbol{x}$ without $\boldsymbol{z}$, mitigating the long-length problem in the reasoning chain.

3.2 Length Generalization on Single Reasoning Tasks

Before introducing our RELAY framework, we first empirically demonstrate the superior length generalization capability of looped Transformers compared to standard auto-regressive models. This analysis serves as the foundation and motivation for our proposed framework.

Experimental Setup. To validate the capabilities of different methods, we use three representative reasoning tasks adapted from Feng et al. (2024) and define problem complexity specific to each task, with full details in Appendix A.

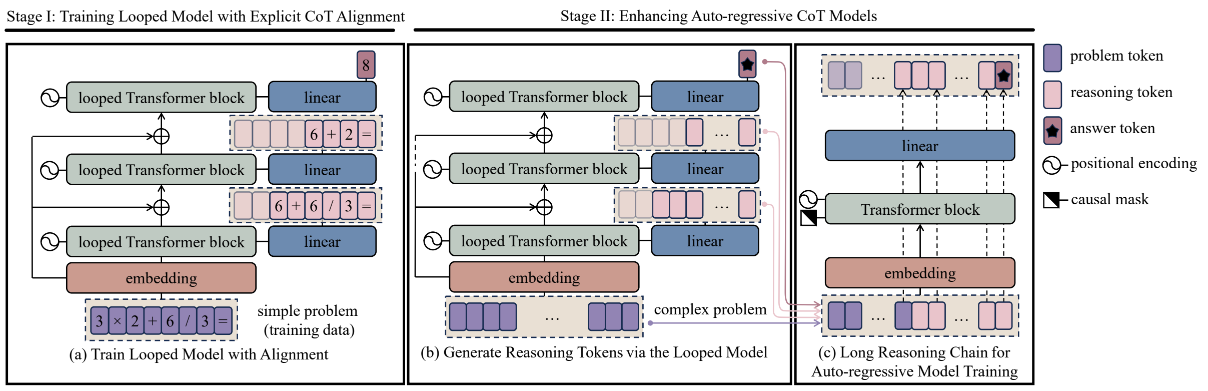
For the auto-regressive CoT model, we employ a standard decoder-only Transformer language model. For the looped model, we use an encoder-only Transformer with bi-directional attention. To address varying problem complexities, we implement dynamic iteration control in the looped model, setting the number of loop iterations equal to the problem complexity. The architectural configuration remains consistent across all models, comprising 3 layers, 256-dimensional hidden states, and 4 attention heads. For positional encoding, we adopt RoPE (Su et al., 2024) across all model variants to enhance sequence encoding for both training and test cases. Performance is evaluated based on the accuracy of the final answer for both models.

Results. Figure 2 illustrates the comparative performance of both models. Within the training distribution (e.g. Arithmetic: ≤ 15 operators), both the looped Transformer and the auto-regressive CoT Transformer achieve perfect accuracy. However, the models exhibit markedly different behaviors when tested on problems exceeding the training length: While the auto-regressive CoT Transformer's performance deteriorates significantly, the looped Transformer maintains superior performance across all length regimes. This demonstrates the length generalization capabilities of the looped Transformer.

3.3 Loop-Enhanced Chain-of-Thought Reasoning

While looped Transformers exhibit superior performance in final answer prediction, they lack interpretability in their intermediate computational processes. Moreover, their design philosophy may struggle with general language tasks, as determining the number of loop iterations becomes challenging beyond reasoning problems. This work seeks to harness the accurate reasoning predictions of the looped model to guide the training of auto-regressive Chain-of-Thought (CoT) models to better handle long-sequence reasoning.

A straightforward way to leverage a well-trained looped model to enhance the auto-regressive CoT model is to use it as a verifier. When a problem is presented, both models generate a final answer, and if the answers match, the CoT output is trusted. However, this approach often fails in practice, as CoT models can produce incorrect reasoning trajectories even when reaching the correct final answer (see Section 4.3), making it unreliable to rely solely on the final answer's accuracy as the guiding signal.



Stage I: Training Looped Model with Explicit CoT Alignment
Stage II: Enhancing Auto-regressive CoT Models
looped Transformer block
linear
embedding
8
6 + 2 =
6 + 6 / 3 =
3 × 2 + 6 / 3 =
simple problem (training data)
(a) Train Looped Model with Alignment
complex problem
(b) Generate Reasoning Tokens via the Looped Model
Transformer block
(c) Long Reasoning Chain for Auto-regressive Model Training
problem token
reasoning token
answer token
positional encoding
causal mask

Figure 3: Overview of the RELAY framework. **Stage I (left):** Training looped model with explicit CoT alignment, where each iteration of the looped model learns to predict corresponding Chain-of-Thought (CoT) steps. **Stage II (right):** Using the trained looped model to generate CoT chains for enhancing auto-regressive CoT models. The looped model generates high-quality CoT chains for complex problems (beyond training length), which are then used to fine-tune the auto-regressive model to improve its reasoning capabilities.

Our key insight is that an alignment can be established between the iterative structure of the looped Transformer and the stepwise nature of CoT reasoning. As shown in Figure 3, unlike the step-by-step token generation in CoT, looped models update their representations simultaneously in each iteration, and the number of such iterations naturally corresponds to the number of reasoning rounds. This structural similarity opens up the possibility of training the looped model to generate the corresponding CoT tokens for each round in parallel, while maintaining its ability to predict the final answer. With this insight, we propose **RELAY** (**<u>RE</u>**asoning through **<u>L</u>**oop **<u>A</u>**lignment iterativel**<u>Y</u>**), a two-stage framework that bridges looped and auto-regressive models.

Stage I: Training Looped Model with Explicit CoT Alignment. In the first stage, we train the looped model to generate intermediate reasoning processes aligned with CoT steps. To formalize this alignment, assume a reasoning chain with T rounds. Given reasoning tokens $\boldsymbol{z} = [z_1, z_2, \dots, z_m]$, denote k_t as the start token position of t-th reasoning round, where each round contains valid reasoning tokens $\boldsymbol{z}_{[k_t:k_{t+1}-1]} = [z_{k_t}, z_{k_t+1}, \dots, z_{k_{t+1}-1}]$. Taking arithmetic reasoning as an example, consider a sequence of tokens representing the complete reasoning chain, "$3\times2+6\div3 = 6+6\div3 = 6+2 = 8$". This sequence can be naturally divided into $T = 3$ rounds using the equal signs as delimiters: Given the input problem "$3 \times 2 + 6 \div 3 =$", the first round corresponds to "$6 + 6 \div 3 =$", the second round corresponds to "$6 + 2 =$", and the third (last) round presents the final answer "8".

Although the number of rounds aligns with the iteration count of the looped model, a key challenge arises from the mismatch in token lengths across different reasoning steps. For instance, earlier steps involving complex expressions (e.g., "$6+6\div3 =$") typically require more tokens than later steps (e.g., "$6 + 2 =$"). This variable length nature poses a challenge for the looped model, which requires fixed-length representations of size n across iterations.

To address this length mismatch while preserving the parallel processing capability of the looped model, we employ a right-aligned padding strategy. For the t-th iteration, we construct a fixed-length sequence $\tilde{\boldsymbol{z}}_t$ of length n by right-aligning the ground truth reasoning tokens $\boldsymbol{z}_{[k_t:k_{t+1}-1]}$ and filling the remaining left positions with <pad> tokens. The fixed-length is determined based on the maximum length among all reasoning rounds and the original input problem (note that the length of a reasoning round usually does not exceed the length of the input problem; otherwise, each round can be further divided into shorter rounds). To track both valid reasoning tokens and the boundary of padding, we introduce a binary mask:

$$M_t[i] = \begin{cases} 1, & \text{if } i = p_t \text{ or } \tilde{\boldsymbol{z}}_t[i] \neq \texttt{<pad>}, \\ 0, & \text{otherwise}, \end{cases} \tag{6}$$

where M_t indicates the positions of valid reasoning tokens and the position of the last <pad> token p_t.

Using this alignment strategy, we train the looped model to predict the corresponding CoT tokens at each iteration, enabling it to generate CoT-aligned intermediate outputs. In detail, at each iteration t, we train the model to predict both the valid reasoning tokens and the last <pad> token through an intermediate prediction head:

$$P(\tilde{\boldsymbol{z}}_t|\boldsymbol{e}_t;\theta_{\text{pred-cot}}), \tag{7}$$

For the intermediate reasoning steps, we ignore all preceding <pad> tokens except the last one, as they have no impact on the reasoning process. The loss of this part can be formulated as :

$$\mathcal{L}_{\text{iter}} = \frac{1}{T}\sum_{t=1}^{T}\text{CrossEntropy}(P(\tilde{\boldsymbol{z}}_t \mid \boldsymbol{e}_t;\theta_{\text{pred-cot}}), \tilde{\boldsymbol{z}}_t) \odot M_t. \tag{8}$$

where the element-wise multiplication $\odot$ ensures that the loss is computed only on valid reasoning tokens and the last <pad> token.

For the final answer, we have the answer prediction loss to ensure correct final predictions:

$$\mathcal{L}_{\text{ans}} = \text{CrossEntropy}(P(\boldsymbol{y}|\boldsymbol{e}_T;\theta_{\text{pred}}), \boldsymbol{y}), \tag{9}$$

where $\boldsymbol{y}$ is the ground truth answer. The total training loss is then:

$$\mathcal{L} = \mathcal{L}_{\text{ans}} + \lambda\mathcal{L}_{\text{iter}}, \tag{10}$$

where λ is a hyperparameter balancing the two objectives.

In our experiments, we find that setting $\lambda = 1$ is already suitable and does not require further tuning. This choice assigns comparable importance to iteration-wise supervision and the final-answer objective, which is also consistent with standard CoT training where each token in the reasoning trace (including the final answer token) is optimized with equal cross-entropy weight.

This design enables the looped model to accurately predict the answer and provide interpretable intermediate reasoning steps that can be effectively utilized to guide the auto-regressive model in Stage II.

Stage II: Enhancing Auto-regressive CoT Models. In the second stage, we leverage the trained looped model to enhance auto-regressive CoT models through a systematic process:

First, we use the trained looped model in Stage I to generate reasoning demonstrations for problems of increasing complexity. For each problem x, we obtain:

$$(\boldsymbol{z}, y) \sim p(\cdot|\boldsymbol{x};\theta_L), \tag{11}$$

where θ_L denotes the trained looped model from Stage I, $\boldsymbol{z} = [z_1, z_2, \ldots, z_m]$ represents the generated reasoning tokens across iterations, and y is the predicted answer.

We then utilize these demonstrations to fine-tune an auto-regressive model. For problem lengths beyond the original training range, we generate a comprehensive dataset of reasoning demonstrations using the looped model. This newly generated data is then merged with the original training dataset, which contains problems within the initial training length. The combined dataset, spanning both the original and extended problem lengths, is then used to fine-tune the auto-regressive model in a single step. This approach allows the model to retain its original reasoning capabilities while acquiring the ability to effectively tackle more complex, longer problems, utilizing the structured insights provided by the demonstrations.

Comparison with Synthetic Data Generation Approaches. To effectively guide the LLMs to handle complex problems, prior works (Hendrycks et al., 2021; Lightman et al., 2024) have explored the synthetic data generation approach, where human labelers construct data generation pipelines based on their understanding of both the task and its solution process. This approach requires labelers to possess comprehensive knowledge in three aspects: (1) problem construction, (2) problem-solving strategies, and (3) pipeline development skills. While effective, this creates a high barrier for deployment across diverse domains, as finding experts who excel in all three areas can be challenging.

In contrast, RELAY reduces these requirements. Our approach follows a more automated pipeline: training data $\rightarrow$ looped model with strong generalization capability $\rightarrow$ longer problem construction $\rightarrow$ automated reasoning generation $\rightarrow$ auto-regressive CoT model training. The human involvement is primarily limited to longer problem construction, eliminating the need for expertise in solution strategies and pipeline development. This reduction in human expertise requirements makes our method more practical and scalable across different domains. Additionally, by leveraging the looped models' inherent generalization capabilities rather than manually designed rules, our approach can potentially capture more nuanced reasoning patterns that might be overlooked in hand-crafted pipelines.

4 Experiments

This section presents a comprehensive empirical evaluation of our RELAY framework through a series of experiments designed to address four key research questions:

- **Q1**: How effectively does the looped model with explicit CoT alignment serve as a general-purpose reasoner across diverse tasks? (Section 4.1)
- **Q2**: What advantages does the looped model with explicit CoT alignment demonstrate in length generalization compared to auto-regressive CoT models? (Section 4.1)
- **Q3**: How can the length generalization capabilities of the looped model with explicit CoT alignment be leveraged to enhance auto-regressive CoT models? (Section 4.2)
- **Q4**: How reliable are the intermediate reasoning steps generated by the looped model with explicit CoT alignment? (Section 4.3)

We address each question through carefully designed experiments, as detailed below.

4.1 Multitask Training

Following the single-task evaluation in Section 3.2, we extend our analysis to a multitask learning setting to examine the general reasoning capabilities of three models: the looped model with explicit CoT alignment, the auto-regressive CoT model, and the vanilla looped model. We jointly train them on three representative reasoning tasks—Arithmetic, Edit Distance (ED), and Longest Increasing Subsequence (LIS)—each requiring multi-step reasoning to reach the correct answer. This setup enables a thorough comparison of the models' ability to generalize effectively across diverse tasks.

For task details, including examples and CoT reasoning steps, please refer to Appendix A.

Experimental Setup. Training datasets follow the same length constraints as in Section 3.2: operator counts ≤ 15 for Arithmetic, input lengths ≤ 30 for ED, and sequence lengths ≤ 100 for LIS. Test datasets extend these ranges to $[15, 25]$, $[30, 40]$, and $[100, 120]$, respectively, to assess length generalization.

Each model is trained jointly on all three tasks using a task-specific token (`[ARI]`, `[ED]`, `[LIS]`) prepended to the input to indicate task type. Evaluation is based on final-answer accuracy under a unified metric.

Results. Figure 4 compares the three models across the three tasks. All models achieve nearly $100\,\%$ accuracy within the training distribution, confirming that the looped model with explicit CoT alignment functions as a general reasoning engine capable of handling diverse multi-step tasks. (**Q1**)

For out-of-distribution cases with longer inputs, both loop-based models markedly outperform the auto-regressive CoT model, highlighting the advantage of iterative architectures for length generalization. Moreover, the looped model with explicit CoT alignment not only retains the strong length generalization capability of the vanilla looped model but further improves upon it, benefiting from the structural correspondence between CoT steps and loop iterations. This alignment strengthens reasoning consistency and interpretability, enabling accurate and explainable reasoning over extended sequences. Overall, the looped model with explicit CoT alignment proves to be a strong and generalizable reasoning framework, surpassing standard auto-regressive CoT models across all tasks. (**Q2**)

4.2 Enhancing Auto-regressive Model with RELAY-Generated CoT Data

In this section, we use the looped model with explicit CoT alignment trained in Section 4.1 to enhance the auto-regressive CoT model through data augmentation. Leveraging its ability to generate accurate reasoning chains for problems beyond the training lengths, we employ these outputs as high-quality data to fine-tune the auto-regressive model.

Experimental Setup. The looped model is employed to generate reasoning chains for extended-length problems—$[15, 25]$, $[30, 40]$, and $[100, 120]$ for Arithmetic, ED, and LIS, respectively. These data are combined with the original training set containing in-range samples. Details of the merged dataset composition are given in Appendix D.2.

We then fine-tune the auto-regressive CoT model on this augmented dataset, starting from the well-trained multitask checkpoint in Section 4.1. The architecture remains unchanged, while weights are updated with the new data to enable longer and more coherent CoT reasoning.

Results. Figure 4 presents the accuracy curves of the RELAY-enhanced auto-regressive CoT model across problem lengths for all three tasks. Compared to the baseline, the model fine-tuned

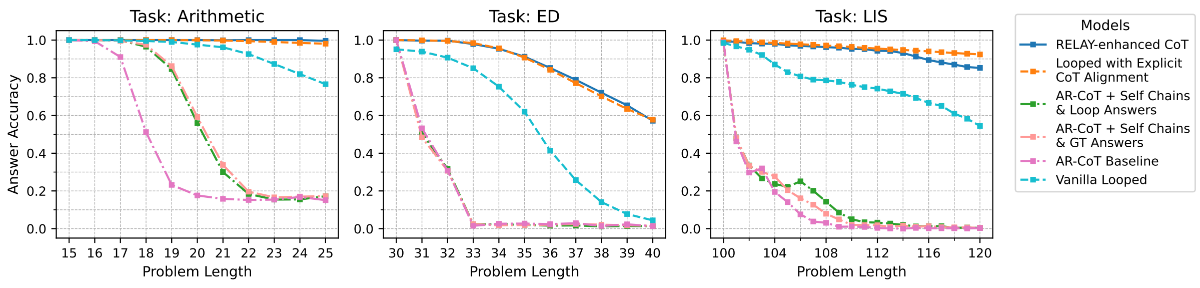


Figure 4: Performance comparison of different models on long reasoning problems across three tasks: Arithmetic, Edit Distance (ED), and Longest Increasing Subsequence (LIS). Accuracy is plotted versus problem length, evaluated beyond the training range. Auto-regressive CoT (AR-CoT) model degrades sharply on longer inputs, while looped models generalize substantially better; explicit CoT alignment yields the strongest extrapolation. Our framework RELAY transfers this advantage to an auto-regressive model, whereas AR-CoT self-training (even with answer filtering) yields only marginal gains.

with RELAY-generated data (from the looped model with explicit CoT alignment) achieves substantial gains on problems beyond the training range. In some cases, its performance even slightly exceeds that of the looped model, while consistently outperforming the baseline auto-regressive model.

These results demonstrate that RELAY effectively transfers the length generalization ability of the looped model to the auto-regressive model through high-quality CoT data generation, enhancing out-of-distribution reasoning without architectural changes. (**Q3**)

4.3 Evaluating the Reliability of RELAY-Generated Intermediate Reasoning Steps

This section examines the reliability of CoT chains generated by the looped model with explicit CoT alignment versus those produced by the auto-regressive CoT model. Specifically, we highlight that the generated data from the auto-regressive CoT model, even when the final result is correct, often contains incorrect intermediate steps, limiting the effectiveness of self-generated data for improving long-sequence performance. In contrast, the looped model produces coherent reasoning chains with both accurate intermediate steps and final outputs, enabling effective fine-tuning and substantial performance gains for the auto-regressive model.

Experimental Setup. We compare data generated by the two models using two metrics: (1) hit matrix and (2) bit accuracy, both designed to assess reasoning reliability.

For the hit matrix, we select the LIS task due to its structured reasoning format. Each reasoning trace forms a $T \times 11$ matrix, where T denotes the number of CoT steps (and loop iterations) and 11 represents the number of tokens per step (10 numbers plus one delimiter <sep>). This structure allows a clear visualization of token-level agreement between generated and ground-truth reasoning steps.

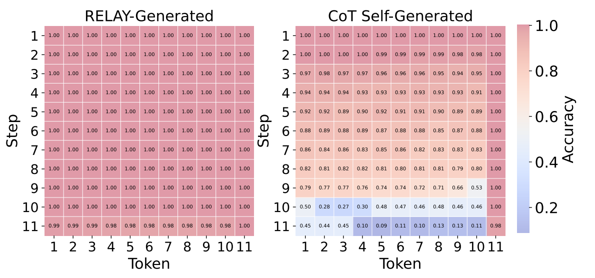


Figure 5: Hit accuracy matrices (step $\times$ token) for LIS with problem length 105 (T=11 reasoning steps). RELAY-generated traces from the looped model with explicit CoT alignment maintain near-perfect token accuracy across all steps, whereas self-generated AR-CoT traces quickly drift, with token accuracy dropping markedly in later steps—indicating substantially higher reasoning-trace quality from the looped generator.

Bit accuracy is computed across all three tasks to quantify token-level correctness within each reasoning step. We compare auto-regressive CoT models fine-tuned with data from each generator type to evaluate the impact of data quality on reasoning consistency.

For the auto-regressive CoT model's self-generated data, we conduct two parallel fine-tuning settings, using either a vanilla looped model or ground-truth answers as verifiers. The procedure is as follows: (1) generate CoT chains for extended-length problems using the auto-regressive model; (2) filter the data with the verifier, retaining only samples whose final answers are correct; and (3) fine-tune the auto-regressive model on the filtered data to enhance its reasoning performance on longer problems.

For comparison, we also fine-tune the same auto-regressive CoT model checkpoint using data generated by the looped model with explicit CoT alignment, under identical fine-tuning parameters and matched sample ratios across different lengths. Both approaches are evaluated across three tasks to assess improvements in reasoning and length generalization.

Results. We evaluate the hit accuracy matrix for the LIS task with a problem length of 105, which corresponds to $T = \lceil 105/10 \rceil = 11$ steps, resulting in an 11×11 matrix (We also provide results for problem length of 101 in Appendix B). As shown in Figure 5, data generated by the looped model with explicit CoT alignment achieves consistently high token accuracy across all positions, with most values approaching 100 %, demonstrating its ability to produce high-quality and reliable data. (**Q4**) In contrast, the data generated by the auto-regressive CoT model exhibits high token accuracy only in the first few positions, while the accuracy steadily decreases in later steps. Although the delimiter tokens `<sep>` at the end of each step achieve high accuracy, this simply implies that the auto-regressive CoT model has only captured the basic format of the reasoning process but fails to predict accurate tokens, which indicates its limited capability to maintain accurate prediction throughout the reasoning process for longer problems.

The bit accuracy results for the models fine-tuned with different datasets across the three tasks (Arithmetic, ED, LIS) and varying problem lengths are provided in Appendix C.

We additionally provide the accuracy of the final answer for the auto-regressive CoT model fine-tuned with self-generated data in Figure 4, noted as "AR-CoT + Self Chains & Loop/GT Answers", corresponding to data filtered by the looped model or ground-truth answers, respectively, which only shows a slight improvement over the baseline model.

5 Conclusion

This paper introduces **RELAY** (**<u>RE</u>**asoning through **<u>L</u>**oop **<u>A</u>**lignment iterativel**<u>Y</u>**), a framework enhancing Chain-of-Thought reasoning by combining looped and auto-regressive Transformers. Our contributions show that (1) a looped Transformer can serve as a general-purpose reasoner with strong length generalization, (2) iteration-wise alignment enables accurate reasoning chain generation beyond training length, and (3) RELAY improves auto-regressive models through high-quality generated reasoning chains. Future work could explore the theoretical foundations of looped Transformers' length generalization and extend RELAY to broader language tasks.

Limitations

Our study focuses on the effectiveness of looped Transformers in structured reasoning tasks, yet their general applicability to broader domains remains uncertain. Determining the optimal number of iterations is also non-trivial, as over- or under-looping may respectively cause redundancy or incomplete reasoning. In addition, step-wise alignment poses challenges due to variable Chain-of-Thought (CoT) lengths, which hinder consistent supervision. Future work will explore how iteration dynamics and CoT variability affect generalization beyond reasoning tasks.

Acknowledgement

DH is supported by National Science Foundation of China (NSFC62376007), National Science Foundation of China (under Key Project No. 92570203) and Beijing Natural Science Foundation (Z250001). This work is supported in part by the State Key Laboratory of General Artificial Intelligence.

References

Bo Chen, Xiaoyu Li, Yingyu Liang, Zhenmei Shi, and Zhao Song. 2024. Bypassing the exponential dependency: Looped transformers efficiently learn in-context by multi-step gradient descent. *arXiv preprint arXiv:2410.11268*.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*.

Ta-Chung Chi, Ting-Han Fan, Peter J Ramadge, and Alexander Rudnicky. 2022. Kerple: Kernelized relative positional embedding for length extrapolation. *Advances in Neural Information Processing Systems*, 35:8386–8399.

Ta-Chung Chi, Ting-Han Fan, Alexander Rudnicky, and Peter Ramadge. 2023. Dissecting transformer length extrapolation via the lens of receptive field analysis. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13522–13537.

Artur Back de Luca and Kimon Fountoulakis. 2024. Simulation of graph algorithms with looped transformers. In *Forty-first International Conference on Machine Learning*.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In *International Conference on Learning Representations*.

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

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. 2024. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36.

Yihang Gao, Chuanyang Zheng, Enze Xie, Han Shi, Tianyang Hu, Yu Li, Michael K Ng, Zhenguo Li, and Zhaoqiang Liu. 2024. On the expressive power of a variant of the looped transformer. *arXiv preprint arXiv:2402.13572*.

Khashayar Gatmiry, Nikunj Saunshi, Sashank J Reddi, Stefanie Jegelka, and Sanjiv Kumar. 2024. Can looped transformers learn to implement multi-step gradient descent for in-context learning? *arXiv preprint arXiv:2410.08292*.

Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. 2023. Looped transformers as programmable computers. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 11398–11442. PMLR.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. 2024. V-STar: Training verifiers for self-taught reasoners. In *First Conference on Language Modeling*.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. 2024. The impact of reasoning step length on large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1830–1842. Association for Computational Linguistics.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Shanda Li, Chong You, Guru Guruganesh, Joshua Ainslie, Santiago Ontanon, Manzil Zaheer, Sumit Sanghai, Yiming Yang, Sanjiv Kumar, and Srinadh Bhojanapalli. 2024. Functional interpolation for relative positions improves long context transformers. In *The Twelfth International Conference on Learning Representations*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Yansheng Mao, Jiaqi Li, Fanxu Meng, Jing Xiong, Zilong Zheng, and Muhan Zhang. 2024. Lift: Improving long context understanding through long input fine-tuning. *arXiv preprint arXiv:2412.13626*.

William Merrill and Ashish Sabharwal. 2024. The expressive power of transformers with chain of thought. In *The Twelfth International Conference on Learning Representations*.

Debjit Paul, Robert West, Antoine Bosselut, and Boi Faltings. 2024. Making reasoning matter: Measuring and improving faithfulness of chain-of-thought reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15012–15032, Miami, Florida, USA. Association for Computational Linguistics.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2024. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*.

Ofir Press, Noah Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Anian Ruoss, Grégoire Delétang, Tim Genewein, Jordi Grau-Moya, Róbert Csordás, Mehdi Bennani, Shane Legg, and Joel Veness. 2023. Randomized positional encodings boost length generalization of transformers. pages 1889–1903, Toronto, Canada. Association for Computational Linguistics.

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

Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron T Parisi, Abhishek Kumar, Alexander A Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Fathy Elsayed, Hanie Sedghi, and 21 others. 2024. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research*. Expert Certification.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomput.*, 568(C).

Yutao Sun, Li Dong, Barun Patra, Shuming Ma, Shaohan Huang, Alon Benhaim, Vishrav Chaudhary, Xia Song, and Furu Wei. 2023. A length-extrapolatable transformer. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. 2024. LESS: Selecting influential data for targeted instruction tuning. In *International Conference on Machine Learning (ICML)*.

Changnan Xiao and Bing Liu. 2023. Conditions for length generalization in learning reasoning skills. *arXiv preprint arXiv:2311.16173*.

Kevin Xu and Issei Sato. 2024. On expressive power of looped transformers: Theoretical analysis and enhancement via timestep encoding. *arXiv preprint arXiv:2410.01405*.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. 2023. Looped transformers are better at learning learning algorithms. *arXiv preprint arXiv:2311.12424*.

Liu Yang, Kangwook Lee, Robert D Nowak, and Dimitris Papailiopoulos. 2024. Looped transformers are better at learning learning algorithms. In *The Twelfth International Conference on Learning Representations*.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2024. Scaling relationship on learning mathematical reasoning with large language models.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. STar: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. LIMA: Less is more for alignment. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Dawei Zhu, Nan Yang, Liang Wang, Yifan Song, Wenhao Wu, Furu Wei, and Sujian Li. 2024. PoSE: Efficient context window extension of LLMs via positional skip-wise training. In *The Twelfth International Conference on Learning Representations*.

A Task Descriptions

We evaluate on three tasks from Feng et al. (2024), including Arithmetic, a mathematical task, and two dynamic programming (DP) problems: Edit Distance (ED) and Longest Increasing Subsequence (LIS). These tasks are chosen for their diverse problem-solving patterns, varying levels of complexity, and their suitability for being solved via a Chain-of-Thought (CoT) reasoning process. Below, we provide detailed descriptions of each task, including input examples, expected answers, and the corresponding Chain-of-Thought (CoT) reasoning steps used to derive the final answers.

1. **Arithmetic**. This task involves computing the answer of arithmetic expressions containing numbers, basic operations $(+, -, \times, \div, =)$, and brackets. Problem complexity is defined as the number of operators. For example:
   - **Input:** $(6 + 9) \div (7 + 2 \times 5 - 4 \times 3) =$
   - **CoT Steps:**

   $$\begin{aligned} &15 \div (7 + 2 \times 5 - 4 \times 3) = \\ &15 \div (7 + 10 - 4 \times 3) = \\ &15 \div (17 - 4 \times 3) = \\ &15 \div (17 - 12) = \\ &15 \div 5 = \end{aligned}$$

   - **Answer:** 3
   - **Problem Complexity:** 6

2. **Edit Distance (ED)**. This task requires computing the minimum number of operations (insert, delete, or replace) needed to transform one sequence into another. The problem complexity corresponds to the length of the shorter string in each pair. The input consists of two sequences separated by a delimiter `|`:
   - **Input:** `o t m l | o t t m l <sep>`
   - **CoT Steps:**

   ```
   0 2 4 6 7 ,
   2 0 2 4 6 ,
   4 2 3 2 4 ,
   6 4 5 4 2 ,
   ```

   - **Answer:** `2`
   - **Problem Complexity:** 4

   Each row corresponds to the edit distance matrix, and the final answer is the edit distance.

3. **Longest Increasing Subsequence (LIS)**. This task identifies the length of longest strictly increasing subsequence in a numerical sequence. We define the problem complexity as $\lceil n/10 \rceil$, where n is the length of the input sequence, as our dataset is structured with 10 numbers per reasoning step. The input is a sequence of integers followed by a delimiter `<sep>`:
   - **Input:** `103 110 145 217 233 18 30 82 141 150 159 161 167 239 <sep>`
   - **CoT Steps:**

   ```
   1 2 3 4 5 1 2 3 4 5 <sep>
   6 7 8 9 9 9 9 9 9 9 <sep>
   ```

   - **Answer:** `9`
   - **Problem Complexity:** $\lceil 14/10 \rceil = 2$

   Here, each CoT step represents an intermediate computation in the dynamic programming process, folded into fixed-size groups (10 numbers per step in our setting) to align with the model structure. If the last group has fewer than 10 numbers, the last number is repeated until the group size reaches 10.

For each task, we construct a dataset consisting of 1 million training samples and 100 k test samples, respectively.

The training datasets are constructed with the length of the problem token sequence $\boldsymbol{x} \leq 15$, 30, and 100 for Arithmetic, ED, and LIS, respectively. To evaluate the model's generalization capabilities, test datasets are created with problem lengths in the ranges $[15, 25]$ for Arithmetic, $[30, 40]$ for ED, and $[100, 120]$ for LIS.

B Hit Matrix for LIS Task with Length 101

We also analyze the hit accuracy matrix for the LIS task with a problem length of 101, corresponding to $T = \lceil 101/10 \rceil = 11$ reasoning steps, as shown in Figure 6. The results exhibit a similar trend to those observed for length 105, with a notable decline in accuracy in the later reasoning steps for data generated by the auto-regressive CoT model, while RELAY-generated data consistently maintains high accuracy. Specifically, while the initial steps maintain relatively high token accuracy, the accuracy deteriorates significantly in later steps, failing to achieve accurate final answers. This highlights a

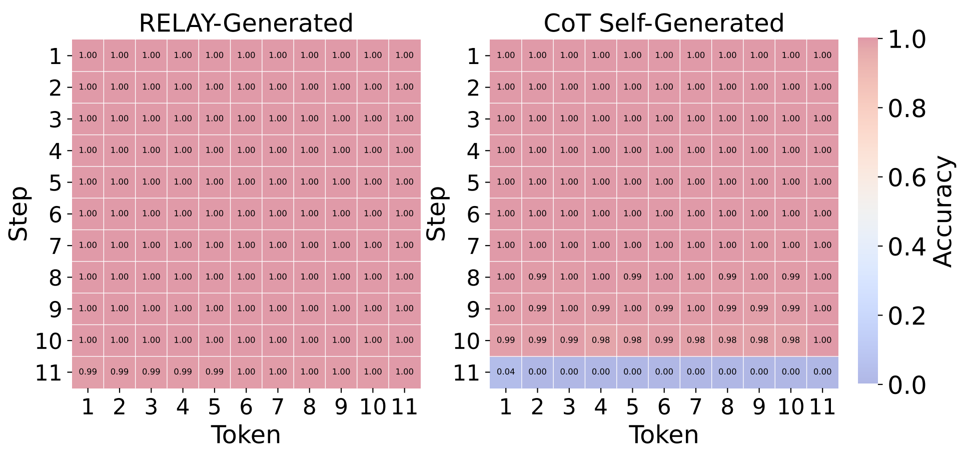


Figure 6: Hit accuracy matrices for the LIS task with a problem length of 101 ($T = 11$ steps). RELAY-generated traces from the looped model stay accurate across steps, whereas self-generated AR-CoT traces accumulate errors over later steps, demonstrating better long-horizon trace quality under RELAY.

key limitation of using CoT self-generated data for supervision: even when the problem length is only slightly beyond the training length, the CoT model struggles to generate accurate reasoning steps towards the end, making it infeasible to fine-tune the model using only the final answer as supervision, as the lack of intermediate reasoning accuracy prevents meaningful improvements in model's performance of handling longer problems.

C Bit Accuracy

The bit accuracy results for the models fine-tuned with different datasets across the three tasks (Arithmetic, ED, LIS) and varying problem lengths are shown in Figure 7. Each subfigure corresponds to one task and includes curves showing the bit accuracy of five models over varying problem lengths: (1) **RELAY-enhanced CoT:** the auto-regressive CoT model fine-tuned with data generated by RELAY, (2) **Looped with Explicit CoT Alignment**, (3) **AR-CoT + Self Chains & Loop Answers:** the auto-regressive CoT model fine-tuned with its self-generated data using looped model answers as labels, (4) **AR-CoT + Self Chains & GT Answers:** the auto-regressive CoT model fine-tuned with its self-generated data using ground-truth answers as labels, and (5) **AR-CoT Baseline:** the baseline auto-regressive CoT model as a reference, also as the initial auto-regressive CoT model before fine-tuned.

As illustrated in Figure 7, both the looped model with CoT alignment and the auto-regressive CoT model fine-tuned with data generated by it consistently achieve high bit accuracy across all tasks and problem lengths. Notably, they maintain over 90 % bit accuracy even at lengths extending beyond the training data (up to $+10$ for Arithmetic and ED tasks, and $+20$ for the LIS task in our setting). This highlights not only the robustness and reliability of the looped model with CoT alignment in length extrapolation scenarios but also the effectiveness of its generated data in significantly enhancing the performance of the auto-regressive CoT model.

In contrast, the auto-regressive CoT model fine-tuned with its own self-generated data shows limited improvement over the baseline model. This is consistent across all tasks and highlights a critical limitation: the self-generated data often contain incorrect intermediate steps, even when the final results are correct. These inaccuracies hinder the model's ability to generalize and perform well on longer problem lengths, reinforcing the importance of reliable intermediate reasoning steps for effective fine-tuning.

D Details for training and fine-tuning

D.1 Implementation Details

In our experiments, we trained three different models: (1) the looped model with CoT alignment, (2) the auto-regressive CoT model, and (3) the vanilla looped model. All models were trained from scratch on the same dataset, which consists of 1 million samples for each of the three tasks. All experiments were conducted on 8 $\times$ NVIDIA RTX 4090 GPUs.

The task-specific training weights and training hyper-parameters are provided in Table 1.

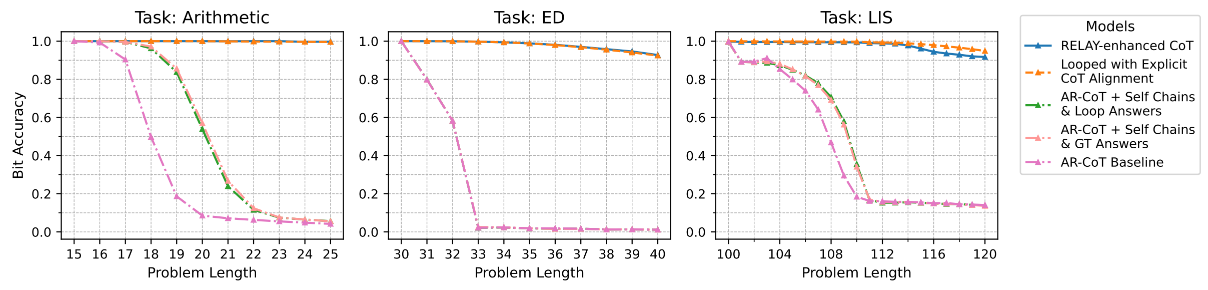


Figure 7: Bit accuracy over varying problem lengths for three tasks: Arithmetic, ED, and LIS. Consistent with the longer setting, RELAY-generated traces remain nearly perfect across problem lengths, while self-generated AR-CoT traces degrade noticeably in longer problems, indicating higher trace fidelity from the looped generator.

Table 1: Training Hyper-parameters of Different Models

Training Hyper-parameters	Looped Model with CoT Alignment	CoT Model	Vanilla Looped Model
Epoch	500	500	500
Batch Size	512	512	512
Learning Rate	5e-4	5e-4	1e-3
Learning Rate Schedule	linear	linear	linear
Warmup Ratio	0.01	0.01	0.01
Optimizer	AdamW	AdamW	AdamW
Weight Decay	0.01	0.01	0.01
Drop out	0.1	0.1	0.1
Weight of ARI	1	1	1
Weight of ED	1	10	10
Weight of LIS	1	5	5

Table 2: Fine-tuning Hyper-parameters

Fine-tuning Hyper-parameters	RELAY-Generated Data	Self-Generated Data
Epoch	500	100 (per phase)
Batch Size	512	512
Learning Rate	1e-4	5e-5
Learning Rate Schedule	linear	linear
Warmup Ratio	0.01	0.01
Optimizer	AdamW	AdamW
Weight Decay	0.01	0.01
Drop out	0.1	0.1
Weight of ARI	1	1
Weight of ED	10	1
Weight of LIS	5	1

For fine-tuning, we used data generated by RELAY and CoT model self-generated data to fine-tune the auto-regressive CoT model. The fine-tuning process followed a similar setup, as detailed in Table 2.

D.2 Sample Length Distribution of Datasets

The original training dataset for training the three models consists of 1 million samples for each task, following a distribution where the number of samples is proportional to problem length.

The merged dataset used for fine-tuning consists of 100 k samples for each of the three tasks, incorporating both the original training data and newly generated samples from extended problem lengths.

For the looped model with explicit CoT alignment, we introduce additional data covering problem lengths of $[16, 25]$ for Arithmetic, $[31, 40]$ for ED, and $[101, 120]$ for LIS. These newly generated samples are merged with the original dataset while maintaining a balanced proportion across different length ranges to ensure effective training. The specific numbers of samples for different problem lengths in the final merged dataset are provided in Table 3.

For the self-generated dataset, we adopt an incremental approach, since the accuracy of original CoT model diminishes rapidly as the problem length increases. Specifically, we maintain a total dataset size of 100 k samples for each task. The initial dataset consists of problems with lengths $\leq 15, 30$, and 100 for Arithmetic, ED, and LIS, respectively. The CoT model is progressively fine-tuned over five phases, each including self-generation on slightly longer problems and followed by 100 epochs of fine-tuning. After each phase, a subset of the current dataset is randomly combined with the newly generated reasoning steps to form an updated synthetic dataset. The maximum number of samples selected for each problem length is detailed in Table 4.

Table 3: Number of Samples for Different Problem Lengths in Merged Dataset Number of training samples per problem length after merging the original in-range data with additional RELAY-generated long-length samples (from the looped model with explicit CoT alignment). This merged dataset is used to fine-tune the auto-regressive CoT model for length generalization.

Task	Arithmetic	ED	LIS
Length	≤ 15	≤ 30	≤ 100
Number of Samples	42515	60844	73235
Length	16	31	101
Number of Samples	6477	4195	1479
Length	17	32	102
Number of Samples	6882	4195	1464
Length	18	33	103
Number of Samples	7287	4055	1449
Length	19	34	104
Number of Samples	6477	4055	1434
Length	20	35	105
Number of Samples	6072	3916	1420
Length	21	36	106
Number of Samples	5668	3916	1405
Length	22	37	107
Number of Samples	5263	3776	1390
Length	23	38	108
Number of Samples	4858	3776	1375
Length	24	39	109
Number of Samples	4453	3636	1360
Length	25	40	110
Number of Samples	4048	3636	1346
Length			111
Number of Samples			1331
Length			112
Number of Samples			1316
Length			113
Number of Samples			1301
Length			114
Number of Samples			1286
Length			115
Number of Samples			1272
Length			116
Number of Samples			1257
Length			117
Number of Samples			1242
Length			118
Number of Samples			1227
Length			119
Number of Samples			1213
Length			120
Number of Samples			1198

Table 4: Number of Samples Generated for Different Lengths For each of the five phases, we list the newly extended length and the maximum number of self-generated samples added for each task. These per-length caps define the incremental synthetic dataset used for AR-CoT self-training while keeping the total dataset size fixed.

Properties		ARI	ED	LIS
Phase I	Length	16	31	101
	Data Generated	15000	15000	15000
Phase II	Length	17	32	102
	Data Generated	10000	10000	10000
Phase III	Length	18	33	103
	Data Generated	7500	7500	7500
Phase IV	Length	19	34	104
	Data Generated	6000	3000	3000
Phase V	Length	20	35	105
	Data Generated	5000	3000	3000