Expediting and Elevating Large Language Model Reasoning via Hidden Chain-of-Thought Decoding

Tianqiao Liu
TAL Education Group
Beijing, China
liutianqiao1@tal.com

Zui Chen
TAL Education Group
Beijing, China
chenzui3@tal.com

Zitao Liu
Jinan University
Guangzhou, China
liuzitao@jnu.edu.cn

Mi Tian
TAL Education Group
Beijing, China
tianmi@tal.com

Weiqi Luo
Jinan University
Guangzhou, China
lwq@jnu.edu.cn

Abstract

Large language models (LLMs) have demonstrated remarkable capabilities in tasks requiring reasoning and multi-step problem-solving through the use of chain-of-thought (CoT) prompting. However, generating the full CoT process results in significantly longer output sequences, leading to increased computational costs and latency during inference. To address this challenge, we propose a novel approach to compress the CoT process through semantic alignment, enabling more efficient decoding while preserving the benefits of CoT reasoning. Our method introduces an auxiliary CoT model that learns to generate and compress the full thought process into a compact special token representation semantically aligned with the original CoT output. This compressed representation is then integrated into the input of the Hidden Chain-of-Thought (HCoT) model. The training process follows a two-stage procedure: First, the CoT model is optimized to generate the compressed token representations aligned with the ground-truth CoT outputs using a contrastive loss. Subsequently, with the CoT model parameters frozen, the HCoT model is fine-tuned to generate accurate subsequent predictions conditioned on the prefix instruction and the compressed CoT representations from the CoT model. Extensive experiments across three challenging domains - mathematical reasoning, agent invocation, and question answering - demonstrate that our semantic compression approach achieves competitive or improved performance compared to the full CoT baseline, while providing significant speedups of at least 1.5x in decoding time. Moreover, incorporating contrastive learning objectives further enhances the quality of the compressed representations, leading to better CoT prompting and improved task accuracy. Our work paves the way for more efficient exploitation of multi-step reasoning capabilities in LLMs across a wide range of applications.

1 Introduction

Chain-of-Thought (CoT) prompting, as introduced by (Wei et al., 2022), involves prompting large language models (LLMs) to generate explicit reasoning steps, significantly enhancing their performance in various reasoning tasks such as mathematical problem solving (Hendrycks et al., 2021, Cobbe et al., 2021) and science question answering (Lu et al., 2022). Subsequent CoT variants (Zhou et al., 2023, Chen et al., 2022, Gao et al., 2023) have aimed at further improving its efficacy across diverse domains. However, these methods enhance LLMs' reasoning capabilities by extending the

Preprint. Under review.

duration and complexity of reasoning processes and incorporating external computational resources to achieve superior outcomes. This increased computational demand may limit their applicability in real-world development scenarios.

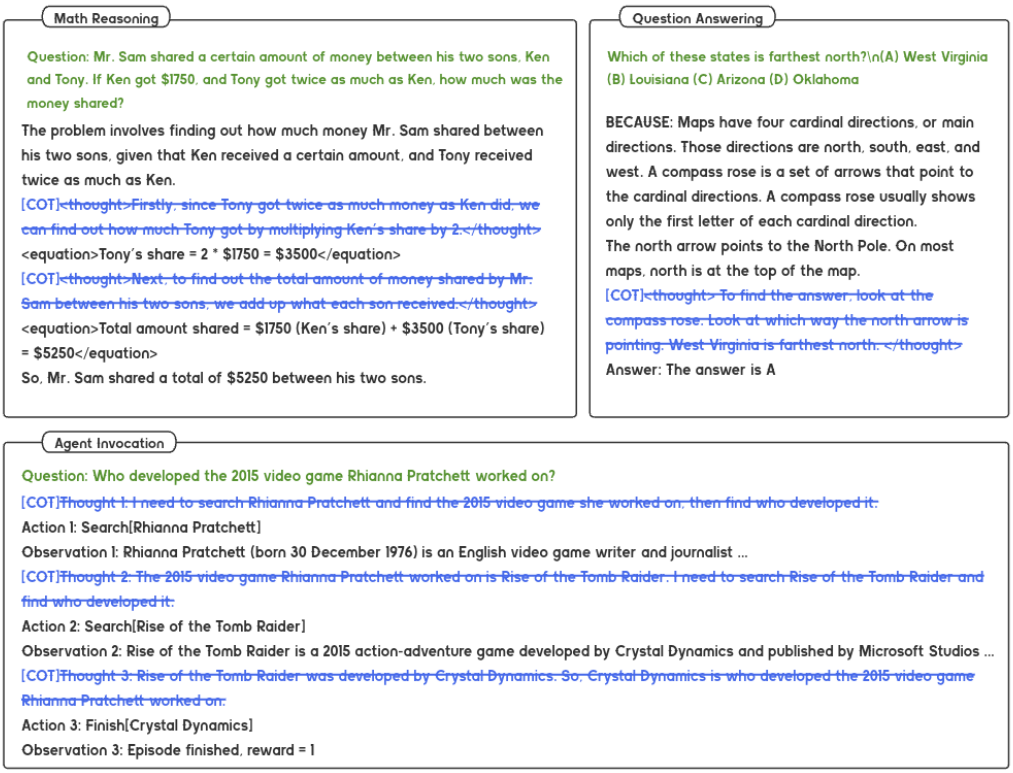

Figure 1: Real-world examples of the CoT prompting in tasks such as mathematical reasoning, question answering, and agent invocation. In the figure, green parts represent actual user queries, blue strikethroughs indicate compressed thought processes, and black text denotes non-CoT content, which corresponds to the expected output for users.

As illustrated in Figure 1, the blue tokens represent intermediate reasoning steps. These steps are crucial for ensuring final decoding accuracy but contribute to significant computational overhead when using standard CoT prompting. Generating complete CoT sequences typically requires substantially longer output sequences. Jin et al. (2024) indicates that extending the reasoning steps in prompts can enhance the reasoning abilities of LLMs, even without introducing new information. Nevertheless, within the transformer architecture (Vaswani et al., 2017), decoding time increases linearly with the output length, leading to higher computational costs and increased latency during inference. These issues, while critical, have not been well addressed in the existing literature.

To address this challenge, we draw upon the human cognitive process, where chains of thought are often implicitly and instantaneously formed within the mind. This introspection led us to hypothesize that during the decoding phase, a single token could represent the forthcoming cognitive process, effectively compressing the semantic content of an extensive reasoning chain into a specialized token. This approach aligns with recent findings in the domain of In-context Learning (ICL) for LLMs. Research by Wang et al. (2023a) has demonstrated the feasibility of employing 'anchor tokens' as potent conduits for aggregating and transmitting complex information.

Leveraging this foundation, we propose a novel two-stage fine-tuning framework aimed at generating subsequent outputs, such as precise answers or computational formulas, by utilizing **a compressed special token** representation in conjunction with the preceding context. The first stage of this framework involves the training of an auxiliary CoT model. This model employs a contrastive loss function to effectively condense an elaborate thought process into a specialized token, herein referred

to as [CoT]. Subsequently, we fine-tune our **Hidden CoT (HCoT) model** to generate the desired output based on the representation of the special token encoded by the CoT model and the preceding instructions, with the parameters of the CoT model remaining frozen. During inference, as shown in Figure 1, our HCoT model halts upon encountering the [CoT] token, at which point it feeds the preceding information to the auxiliary CoT compression model. The auxiliary model then generates a compressed CoT representation encapsulating the subsequent thought process. This compressed representation is then reinserted into the HCoT model to complete the inference. Concurrently, the auxiliary CoT compression model can either continue to generate the full thought process or opt not to, in order to conserve computational resources. Building on the inherent parallelizability of the LLM encoding process, the encoding phase that yields the special token representation is markedly more time-efficient when compared to the time-consuming process of decoding a complete chain of thought. Consequently, this optimization significantly accelerates the rate of inference.

Our extensive experiments show the potential of HCoT method on four datasets in three challenging domains: mathematical reasoning (Hendrycks et al., 2021, Cobbe et al., 2021), agent invocation (Yao et al., 2023), and science question answering (Lu et al., 2022). The results demonstrate that our HCoT model achieves competitive or improved performance compared to the full CoT baseline, while providing significant speedups of over 1.5x to 3.8x in decoding time.

To summarize, our major contributions are:

- We propose the Hidden Chain-of-Thought (HCoT) framework, a novel approach that accelerates the inference process of large language models by compressing the multi-step reasoning process into a specialized token representation, thereby reducing computational overhead during decoding.
- We introduce a disentangled training paradigm for the multi-step CoT reasoning, enabling isolated error correction and specialized optimization for each component.
- Our compression model effectively condenses the entire thought process into a compact special token while maintaining interpretability, allowing for parallel generation of CoT content.
- By incorporating a contrastive learning objective, we further enhance the quality of the compressed CoT model. This approach improves CoT prompting and task accuracy through the application of a span-level loss function during supervised fine-tuning.

We believe our work paves the way for more efficient exploitation of multi-step reasoning capabilities in LLMs across a wide range of applications.

2 Background

We first formalize some existing methods in this section. Our approach is not only inspired by these prior techniques but is also benchmarked against them for comparative analysis. We denote a pre-trained LLM with parameters θ. We use lowercase letters such as x, c, and z to represent the user's question, the generated content, and the CoT reasoning process, respectively. For instance, a user question is denoted as $x = (x[1], \cdots, x[n])$, where each $x[i]$ is an individual token, and the probability of the sequence under our model is given by $p_\theta(x) = \prod_{i=1}^{n} p_\theta(x[i]|x[1]...x[i-1])$. To accommodate the complexity of reasoning that involves multiple interleaved sequences of content and thought, we extend our notation. For the *i*-th element in the reasoning process, we use subscripts: z_i represents the *i*-th chain of thought, and c_i represents the *i*-th output content sequence. We use uppercase letters C and Z to denote a collection of output contents and thoughts respectively.

Chain-of-Thought (CoT) Reasoning introduces intermediate steps that guide the model towards generating a more structured and potentially more accurate response. In this framework, the output is divided into several components: intermediate steps z_i and content parts c_i. The process involves iteratively generating these components based on previous steps: where model sample *i*-th thought $z_i \sim p_\theta(z_i \mid x, c_0, z_0 \cdots c_i)$ and then sample following content $c_{i+1} \sim p_\theta(c_{i+1} \mid x, c_0, z_0 \cdots c_i, z_i)$ given the sampled *i*-th thought.

Reasoning w/o Chain-of-Thought (CoT) contrasts with the CoT methodology by directly generating the final answer without explicitly modeling the intermediate reasoning steps. In this approach, the model aims to produce the output content C directly from the user's question x: $C \sim p_\theta(C \mid x)$. Here, the the reasoning process is implicit within the model's parameters θ and is not visible.

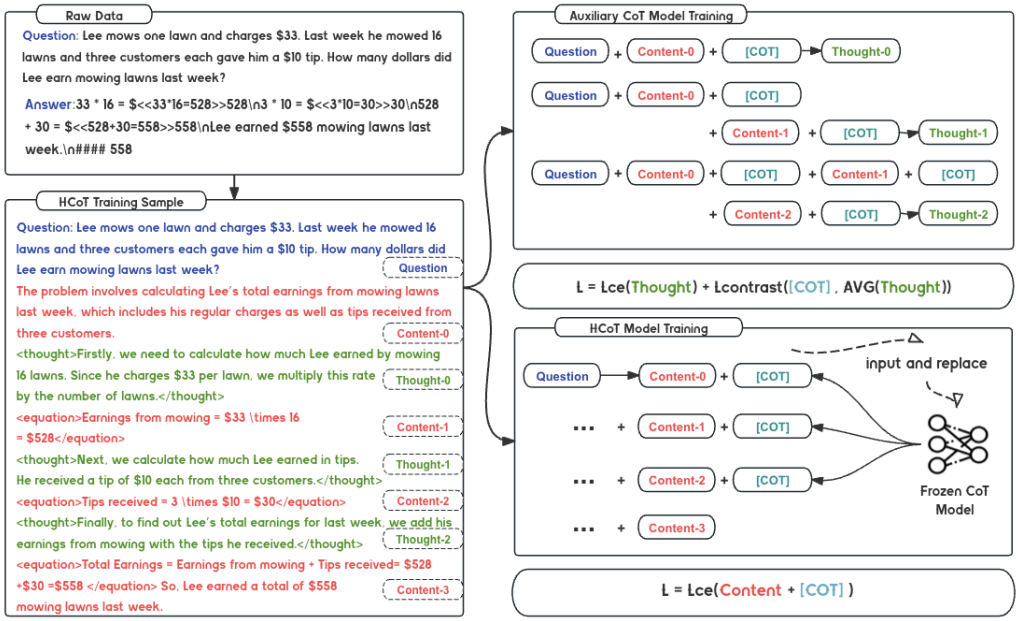

Figure 2: Data construction and two-stage training of Hidden Chain-of-Thought (HCoT) models for math reasoning tasks: Training instances are synthetically generated from raw data using GPT-4, then utilized separately for training the Auxiliary Chain-of-Thought Model and the HCoT Model.

3 HCoT: Hidden Chain-of-Thought Reasoning

In this section, we present our novel two-stage training method that incrementally develops the auxiliary CoT model followed by the Hidden CoT (HCoT) model. The auxiliary CoT model is ingeniously crafted to distill the reasoning process into a singular `[CoT]` token. Subsequently, the HCoT model leverages the encapsulated reasoning within the `[CoT]` token to facilitate swift and efficient chain-of-thought reasoning. As depicted in Figure 2, our training paradigm encompasses three components: (i) the generation of HCoT training instances from the original dataset to construct data that embodies CoT reasoning, (ii) the auxiliary CoT model training that employs these HCoT instances to create specialized training samples aimed at enhancing CoT reasoning, and (iii) the HCoT Model training phase that utilizes the same HCoT training instances, first replacing the intermediate reasoning steps z_i with the special `[CoT]` token, and upon encountering this special token, leveraging the encoded hidden representation from the frozen Auxiliary CoT model to replace the original input embedding for the special `[CoT]` token in the HCoT Model, thereby guiding the generation of the subsequent content c_{i+1}. In the following subsections, we delineate the methodology for constructing HCoT training samples in Section 3.1, elucidate the training dataset preparation and the training procedure for the auxiliary CoT model in Section 3.3, and detail the dataset construction and training process for the HCoT model in Section 3.4.

3.1 HCoT Training Sample Construction

Constructing training samples for HCoT is a flexible process that can be tailored to the specific requirements of the task at hand and the anticipated format of the CoT. For instance, in our experiments on math reasoning tasks, as depicted in Figure 2, data from sources such as Math and GSM8K are utilized to construct training samples employing a GPT-4 based ICL method. (Specific prompts are in AppendixB.) This approach enables the model to output a series of contents and thoughts denoted as $c_0, z_0, \ldots, z_{n-1}, c_n$ which is sampled from:

$$p(C, Z \mid x) = \prod_{i=1}^{n} p_\theta(c_0 \mid x) p_\theta(z_{i-1} \mid x, \ldots, c_{i-1}) p_\theta(c_i \mid x, c_0, \ldots, z_{i-1}), \tag{1}$$

This recursive formulation encapsulates the dependencies between the **intermediate reasoning steps** (z_{i-1}) and the content (c_i) that is produced as a result. The flexibility of the HCoT sample construction process allows for the adaptation of the training regime to accommodate diverse reasoning patterns and content structures. By iteratively generating and conditioning on the chain of thought, the model is trained to better align its reasoning process with the underlying logic of the task.

3.2 Disentangled Training Paradigm

Delving deeper into Equation 1, we have disentangled the probability distribution into distinct components, we can identify the segment $p_\theta(z_{i-1} \mid x, \ldots, c_{i-1})$ as the generation phase of CoT information, namely **auxiliary CoT model**. Conversely, the segment $p_\theta(c_i \mid x, c_0, \ldots, z_{i-1})$ is responsible for harnessing the generated CoT to produce subsequent content, namely content generation model. This observation has led to the conceptualization of disentangled training paradigm which first compressing the high-dimensional discrete distribution of $p_\theta(z_{i-1} \mid x, \ldots, c_{i-1})$ into a more compact thought representation via encoding the input content with p_θ^{COT}, which corresponding to the auxiliary CoT model trianing. Subsequently, the content generation model p_θ^{HCoT} is fine-tuned to maximize $p(c_i \mid x, c_0, \ldots, z_{i-1})$, where $[z_0 \ldots z_{i-1}]$ denote the compressed representations provided by the CoT model. This disentangled training paradigm offers several beneficial training dynamics:

Error Isolation: By decoupling the training of the auxiliary CoT model (p_θ^{COT}) from the content generation model (p_θ^{HCoT}), errors in reasoning can be isolated within the auxiliary CoT model. This facilitates targeted corrections without affecting the content generation model, thereby preventing the propagation of errors and enhancing the overall robustness of the system.

Specialized Optimization: The disentangled training paradigm allows for specialized optimization strategies. The CoT model can be honed to refine reasoning abilities and logical coherence, while the content generation model can concentrate on articulating clear and pertinent content. This specialization ensures that each model maximizes its performance in its respective domain, leading to a more effective training process.

Parallel Development and Improved Interpretability: The CoT generation operates in parallel with the generation of actual content. This not only accelerates the inference speed but also preserves the interpretability of the model. Unlike a black box approach, the reasoning steps generated by the CoT model are explicit and can be scrutinized, allowing for a better understanding of the model's thought process and facilitating easier debugging and refinement.

To be noticed, We also include the part of $p_\theta(c_0 \mid x)$ in our content generation model.

3.3 Auxiliary CoT Model

In this section, We use lowercase letter r to represent special `[CoT]` token's representation, and r_i denotes the i-th special `[CoT]` representation. Given the user's question and the preceding content, the objective of the auxiliary CoT model is to distill the reasoning process into a compact representation by maximizing $p_\theta^{COT}(z_i \mid x, \ldots, c_i, r_i)$, where z_i is the desired thought process.

Training Data Configuration: As depicted in the top right corner of Figure 2, the training data for the auxiliary CoT model is constructed by first extracting all the thought processes from the original HCoT training samples. Subsequently, between each content segment c_i and each thought segment z_i, we insert a special token, denoted as `[CoT]`, where this unique token serves as an anchor to facilitate the generation of the subsequent thought process. We then segment the entire training sample into individual instances, each treating a thought process z_i as the target output, conditioned on the preceding context comprising the question x, special `[CoT]` tokens and the content segments up to c_{i-1}.

Thought Compression: The auxiliary CoT model p_θ^{COT} is trained by maximizing the likelihood $p_\theta^{COT}(z_i \mid x, \ldots, c_i, r_i)$, where we expect the model to generate the most accurate thought representation based on the preceding information. In addition to the conventional cross-entropy loss, we incorporate symmetric contrastive loss between **the thought process respresentations mean-pooling** and the [CoT] token representation to enhance the thought compression capability of the model. The underlying assumption is that the compressed thought representation should exhibit a higher affinity with its corresponding special `[CoT]` token than with other `[CoT]` tokens, and vice versa. The final loss function for the auxiliary CoT model is as follows:

$$\mathcal{L}_{\text{CoT}} = \mathcal{L}_{\text{CE}} + \lambda \cdot \mathcal{L}_{\text{contrastive}} = -\log p_\theta^{CoT}(z_i \mid x, \ldots, c_i, r_i) - \frac{\lambda}{2} \cdot (log \frac{exp(\mathbf{z}_i \cdot \mathbf{r}_i)}{\sum_{k=0}^{n} exp(\mathbf{z}_i \cdot \mathbf{r}_k)} + log \frac{exp(\mathbf{z}_i \cdot \mathbf{r}_i)}{\sum_{j=0}^{n} exp(\mathbf{z}_j \cdot \mathbf{r}_i)}) \tag{2}$$

, where n denotes the batch size during the auxiliary CoT model's training, $\mathbf{z}_i \in \mathcal{R}^d$ represents the normalized representation of the i-th target thought process, obtained by mean-pooling the final hidden states from $p\theta^{COT}$ when the input is the sequence $[z_i[0], z_i[1], \ldots, z_i[t]]$. Additionally, $\mathbf{r}_i \in \mathcal{R}^d$ denotes the normalized representation of the corresponding special `[CoT]` token generated by the auxiliary CoT model. The $\mathcal{L}_{\text{contrastive}}$ is introduced to enhance the compactness of the thought process representation, ensuring that it exhibits a higher affinity towards the corresponding target thought process representation. Here, λ is a hyperparameter that governs the trade-off between the contrastive loss and the primary cross-entropy loss term.

3.4 HCoT Model

Training Data Configuration: The training data construction process for the HCoT model is illustrated in the bottom right corner of Figure 2. We replace all the thought processes z_i in the original HCoT training samples with the special `[CoT]` token. This transformation aligns the training paradigm seamlessly with the auxiliary CoT model's training, as they share the same input format. By substituting the explicit thought processes with the compact `[CoT]` token, we effectively leverage the distilled reasoning encapsulated within the auxiliary CoT model's output representation. This approach ensures that the HCoT model's training is closely tied to the learned thought representations from the auxiliary CoT model, facilitating the transfer of reasoning capabilities.

Supervised Fine-tuning of HCoT Model: Given the user's question x, the objective of the HCoT model is to maximize $\prod_{i=1}^{n} p_\theta^{HCoT}(c_0 \mid x) p_\theta^{HCoT}(c_i \mid x, c_0, \ldots, \mathbf{z}_{i-1})$, where $\mathbf{z}_{i-1}$ is the compressed thought representation obtained from the auxiliary CoT model. In this stage, we freeze the parameters of the auxiliary CoT model and fine-tune the HCoT model to effectively leverage the reasoning encapsulated within the compressed representations. Notably, the training target sequence comprises both content segments and special `[CoT]` tokens, requiring the model to learn not only the generation of content but also the appropriate insertion of the compressed thought representations denoted by the `[CoT]` tokens. We employ the standard cross-entropy loss function to supervise the fine-tuning process of the HCoT model.

4 Experiment

4.1 Datasets

We conducted experiments on three downstream tasks including four seed datasets: GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), ScienceQA (Lu et al., 2022), and HotpotQA (Yang et al., 2018). The ScienceQA dataset was categorized into three subjects: ScienceQA (natural science), ScienceQA (social science), and ScienceQA (language science). We prepared the training, validation, and test data for the auxiliary CoT model and the HCoT model based on the seed datasets, as detailed in Table 3. It is worth noting that there is no dedicated test data for the auxiliary CoT model, as we evaluate the end-to-end performance of the HCoT model. During training, we select the best-performing auxiliary CoT model by identifying the model with the lowest perplexity score on the auxiliary CoT model validation set. The train, validation, and test datasets remain consistent across other baselines for fair comparison. Notably, the auxiliary CoT training and validation data were constructed from the HCoT training and validation portions of the data, preventing the usage of more data than other baseline methods.

For the math reasoning datasets, GSM8K and MATH, we generated separate fields for thoughts and contents based on the input questions using the method described in Section 3.1, implemented by GPT-4. For the question answering task, we directly utilized the original fields from the ScienceQA dataset[1]. Notably, for the agent invocation task, we employed fields generated through the ReAct[2]

[1] `https://scienceqa.github.io/`

[2] `https://react-lm.github.io/`

framework for the HotpotQA dataset, treating the final answer path as the target. The final HCoT training samples can be referred to in Figure 1. It is important to note that due to some data instances containing multiple CoT tokens, there is a discrepancy in the number of training data entries between the auxiliary CoT model and the HCoT model. Moreover, for the ScienceQA dataset, we focused on the subset of questions that did not involve image understanding, and for the HotpotQA dataset, we concentrated on the training samples that achieved the correct final answer with ReAct, resulting in differences from the original dataset sizes.

4.2 Experimental Setup

To assess the efficacy of our method, we selected two base models for comparison: LLaMa2-7B and LLaMa2-13B (Touvron et al., 2023). We trained separate models for each domain, and we selected the best-performing checkpoint based on its performance on the corresponding development set and evaluated the final results using the test portion described in Section 4.1. Specifically, the math domain HCoT model was trained by combining the training data from both the GSM8K and MATH datasets. For all tasks, we employed accuracy as the primary evaluation metric. In the math reasoning domain, accuracy represents the proportion of questions answered correctly. For the ScienceQA dataset, accuracy corresponds to the correct selection among the multiple-choice options (A, B, C, D). For the agent invocation task on the HotpotQA dataset, we first identified the samples with the correct final answer and considered all actions along the path to be correct explorations. Subsequently, we calculated the ratio of correctly invoked actions as the agent invocation accuracy. More implementation details are provided in the Appendix D.5.

4.3 Baselines

To comprehensively study our method, we conducted experiments under five different settings:

- Zero/Few-shot CoT: Applied zero-shot CoT prompting on MATH and GSM8K datasets, and few-shot prompting on others, as a reference for models' inherent capabilities without task-specific training.
- Train without COT: Removed thought processes and trained on remaining content in Figure 2, establishing a baseline without CoT reasoning.
- Train with COT: One-stage training on data without removing thoughts, serving as a baseline with explicit CoT reasoning.
- Train with HCoT base: Two-stage training (Figure 2) with cross-entropy loss for the auxiliary CoT model.
- Train with HCoT Contrast: Complete two-stage training with contrastive loss for the auxiliary CoT model, representing the final proposed method.

4.4 Results

The experimental results presented in Table 1 provide valuable insights into the effectiveness of our proposed HCoT approach across various tasks and datasets, in comparison with baseline methods. Overall, we observe performance improvements with the HCoT approach under most experimental settings. The HCoT models achieved the top performances in most cases, except for the social science question answering task in the Science QA dataset. In general, training with the Chain of Thought (CoT) technique outperformed the baseline without CoT, and further improvements were observed when training with the proposed HCoT approach. Notably, in the agent invocation task evaluated on the HotpotQA dataset, the HCoT-Contrast method exhibited significant improvements, with accuracy gains of 1.21% and 1.96% for the LLaMa2-7B and LLaMa2-13B models, respectively, compared to the CoT training baseline. Furthermore, for the question answering task represented by the ScienceQA dataset, the HCoT-Contrast approach demonstrated superiority in the natural science and language science subsets. In the natural science subset, the performance gains were 3.25% and 0.18% for the LLaMa2-7B and LLaMa2-13B models, respectively, compared with the CoT training. Similarly, in the language science subset, the improvements were 1.72% and 0.55%. In the math reasoning task, the training with HCoT achieved the best performance under both the 7B and 13B settings. These results highlight the efficacy of our proposed method in enhancing reasoning capabilities across various tasks and datasets.

Table 1: Performance comparison of LLaMa2-7B and LLaMa2-13B models under various training scenarios for different tasks and datasets. The ScienceQA dataset is further divided into three subsets: Natural Science (NS), Social Science (SS), and Language Science (LS), with separate results reported for each subset. The top performances in each category and model size are highlighted in bold.

	Models	Math		Science QA			Agent Invoke
		GSM8K	MATH	NS	SS	LS	HotpotQA
Zero/Few CoT	LLaMa2-7B	14.60	2.50	56.80	51.64	67.06	47.11
	LLaMa2-13B	28.7	3.90	58.98	48.42	67.09	51.74
Train w/o CoT	LLaMa2-7B	34.27	6.80	82.10	62.32	87.36	79.78
	LLaMa2-13B	42.15	9.44	84.72	66.14	88.18	82.36
Train w CoT	LLaMa2-7B	36.85	6.74	80.99	**65.8**	86.64	83.73
	LLaMa2-13B	43.97	10.16	84.46	**69.74**	89.36	83.89
Train HCoT	LLaMa2-7B	**37.15**	7.49	83.13	63.89	**88.45**	83.5
	LLaMa2-13B	43.82	10.68	84.41	68.84	89.27	85.81
Train HCoT-Contrast	LLaMa2-7B	36.47	**8.24**	**84.24**	62.77	88.36	**84.94**
	LLaMa2-13B	**44.43**	**11.16**	**84.64**	68.5	**89.91**	**85.85**
δ w HCoT	LLaMa2-7B	-0.38	1.5	3.25	-3.03	1.72	1.21
	LLaMa2-13B	0.46	1.00	0.18	-1.24	0.55	1.96
δ wo Contrast	LLaMa2-7B	0.68	-0.75	-1.11	1.12	0.09	-1.44
	LLaMa2-13B	-0.61	-0.48	-0.23	0.34	-0.64	-0.04

Effect of HCoT Framework: The results presented in the row "δ w HCoT" highlight the impact of our proposed HCoT framework in comparison to the full CoT training approach. Across most tasks and datasets, we observe performance gains when employing the HCoT framework, as indicated by positive values in this row. For the LLaMa2-7B model, the HCoT framework led to substantial improvements in the natural science (3.25%) and language science (1.72%) subsets of the ScienceQA dataset, as well as in the agent invocation task on the HotpotQA dataset (1.21%). However, slight performance decreases were observed in the GSM8K (-0.38%) and social science (-3.03%) tasks. Similarly, for the larger LLaMa2-13B model, the HCoT framework demonstrated its effectiveness, yielding notable performance gains in the agent invocation task on the HotpotQA dataset (1.96%), as well as improvements in the GSM8K (0.46%), MATH (1.00%), and language science (0.55%) tasks. Modest decreases were observed in the social science subset (-1.24%).

Effect of Contrstive Learning: The results presented in the row "δ wo Contrast" highlight highlights the impact of incorporating contrastive learning into our HCoT framework. Negative values in this row indicate performance decreases when the contrastive loss objective is not employed, suggesting the effectiveness of contrastive learning in enhancing the model's reasoning capabilities. The predominance of negative values in the "δ wo Contrast" row across various tasks and datasets highlights the significant role of contrastive learning in our HCoT framework. By incorporating contrastive loss, the model's ability to effectively capture and leverage the core reasoning steps is enhanced, leading to improved performance in most cases compared to the HCoT approach without contrastive learning.

4.5 Discussion

The Speedup of HCoT Model: Table 2 presents the compression and speedup rates of the HCoT model during the inference stage across four datasets compared to the explicit CoT model, both of which are based on LLaMa2-7B. The results for the LLaMa2-13B model are included in the Appendix due to space limit. Firstly, let's clarify the metrics used in the table. S-CR (Sequence-Level Compression Rate) refers to the average number of completion tokens of the HCoT model compared to the CoT model. S-S (Sequence-Level Speedup) is the reciprocal of S-CR, representing how many times faster the HCoT model is compared to the Full CoT model. W-CR (Wall-clock Time Compression Rate) provides a realistic measure of user-perceived speed-up, by comparing the actual inference time of the HCoT model to the Full CoT model. W-S (Wall-clock Time Speedup) is the reciprocal of W-CR. The table shows that S-CR values range from 23.78% to 66.91%, indicating that the sequence length of the HCoT model's output is significantly shorter than that of the Full CoT model. Correspondingly, the W-CR values range from 35.82% to 71.04%, and W-S values

range from 1.41x to 2.79x, showcasing a substantial acceleration. It's important to note that the sequence-length-based acceleration rate (S-S) is generally higher than the real-time acceleration rate (W-S). This discrepancy arises because the sequence-length-based measure does not account for the encoding time of the Auxiliary CoT Model in HCoT. These times were tested on an H800 cluster using a single 80GB GPU. Despite achieving a speedup range of 1.41x to 2.79x, the HCoT model also delivers superior performance compared to the Full CoT model, demonstrating its efficiency and effectiveness. More fine-grained analysis of the length distribution before and after compression are detailed in Appendix.

The recovery of CoT process: Although in general scenarios, people prefer concise yet accurate responses that have undergone CoT reasoning, it is reasonable to compress the CoT process simply at this time. However, in certain situations, people wish to see the complete CoT process, thus maintaining the option to retain full output of CoT is important. Our model employs a method similar to "hiding", where the complete CoT process is concealed during the main reasoning process of the HCoT model, but the Auxiliary CoT model can still produce it normally when required. When asked to display the complete CoT process, we can simply keep the other settings unchanged and request the CoT model to continue outputting as needed. The case study in Appendix D shows this process.

Table 2: Compression and Speedup Rates of the HCoT model during inference across four datasets compared to Full CoT Model.

LLaMa2-7B				
Task	GSM8K	MATH	ScienceQA	HotpotQA
S-CR	60.45%	55.72%	66.91%	23.78%
S-S	1.65x	1.79x	1.49x	4.21x
W-CR	62.48%	62.49%	71.04%	35.82%
W-S	1.60x	1.60x	1.41x	2.79x

5 Related Work

Chain-of-thought prompting (CoT) enhances the emergent reasoning abilities of LLMs by prompting them to use explicit reasoning steps. Zero-shot-CoT (Kojima et al., 2023) demonstrates notable improvements in diverse reasoning tasks by merely prefacing solutions with the phrase "Let's think step by step,". The least-to-most prompting (Zhou et al., 2023) approach effectively addresses complex problems by decomposing them into manageable subproblems and resolving them sequentially. Wang et al. (2023b) considers the self-consistency of CoT, enhancing its performance through majority voting. Furthermore, Gou et al. (2023) and Yuan et al. (2023) employ CoT on GPT-4, stabilizing the CoT capabilities of open-source models through fine-tuning on sampled data. Recent studies like **?** and **?** highlight the impact of reasoning step length on performance, suggesting that extending reasoning steps can improve outcomes. Our method achieves a similar effect by effectively extending CoT reasoning length, but with the added benefit of saving time during the decoding phase.

Efficient model inference often utilizes model compression techniques (Han et al., 2016) such as pruning or quantization. LLM-Pruner (Ma et al., 2023) implements structural pruning, which selectively removes non-critical coupled structures based on gradient information. LLM-QAT (Liu et al., 2023) leverages generations produced by the pre-trained model, enabling quantization of any generative model independently of its training data, akin to post-training quantization methods. Additionally, Gloeckle et al. (2024) considers multi-token prediction as an auxiliary training task, asking the model to predict the following n tokens using n independent output heads at each position in the training corpus, which not only accelerates inference but also enhances performance. **?** proposes compressing prompts with gist tokens, focusing on efficient encoding. Deng et al. (2023) is closely related to ours, employing a method where the model is trained to predict hidden states for implicit CoT reasoning, aiming to reason more effectively. However, this approach exhibits a significant performance decline compared to explicit CoT reasoning, lacks interpretability, and involves a more complicated training process. In contrast, our method streamlines training, enhances reasoning path optimization, and sustains robust performance across tasks using models with over 7B parameters, offering a more interpretable, practical, and efficient solution.

6 Conclusion

We proposed HCoT, an innovative framework designed to accelerate the inference process of large language models while preserving their multi-step reasoning capabilities. At its core, HCoT employs a disentangled training paradigm that decouples the reasoning process into two specialized components: an auxiliary CoT model and a content generation model. The auxiliary CoT model is trained to compress the entire thought process into a compact, specialized token representation through a contrastive loss objective. This compressed representation effectively encapsulates the core reasoning steps, enabling efficient parallel computation during inference. The HCoT model, in turn, is fine-tuned to leverage this compressed reasoning representation, seamlessly integrating it into the content generation process. Our extensive experiments across diverse domains demonstrate the efficacy of the proposed HCoT framework. The results highlight its ability to achieve competitive or improved performance compared to the full CoT baseline while providing significant speedups of at least 1.5x in decoding time. However, it is important to acknowledge that this increase in efficiency comes at the cost of a more complex training phase and the necessity for additional model parameters, which may present scalability and resource challenges. In the future, we hope to address these limitations by optimizing the training phase and enhancing the model's scalability, potentially reducing the need for additional parameters and lessening the training resource burden.

References

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. 2023. Implicit Chain of Thought Reasoning via Knowledge Distillation. ArXiv:2311.01460 [cs].

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. 2024. Better & Faster Large Language Models via Multi-token Prediction. ArXiv:2404.19737 [cs].

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Tora: A tool-integrated reasoning agent for mathematical problem solving.

Song Han, Huizi Mao, and William J. Dally. 2016. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. ArXiv:1510.00149 [cs].

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.

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. ArXiv:2401.04925 [cs].

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large Language Models are Zero-Shot Reasoners. ArXiv:2205.11916 [cs].

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2023. LLM-QAT: Data-Free Quantization Aware Training for Large Language Models. ArXiv:2305.17888 [cs].

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *The 36th Conference on Neural Information Processing Systems (NeurIPS)*.

Training Large Language Models to Reason in a Continuous Latent Space

Shibo Hao[1,2,*], **Sainbayar Sukhbaatar**[1], **DiJia Su**[1], **Xian Li**[1], **Zhiting Hu**[2], **Jason Weston**[1], **Yuandong Tian**[1]

[1]FAIR at Meta, [2]UC San Diego
[*]Work done at Meta

Large language models (LLMs) are restricted to reason in the "language space", where they typically express the reasoning process with a chain-of-thought (CoT) to solve a complex reasoning problem. However, we argue that language space may not always be optimal for reasoning. For example, most word tokens are primarily for textual coherence and not essential for reasoning, while some critical tokens require complex planning and pose huge challenges to LLMs. To explore the potential of LLM reasoning in an unrestricted latent space instead of using natural language, we introduce a new paradigm COCONUT (Chain of Continuous Thought). We utilize the last hidden state of the LLM as a representation of the reasoning state (termed "continuous thought"). Rather than decoding this into a word token, we feed it back to the LLM as the subsequent input embedding directly in the continuous space. Experiments show that COCONUT can effectively augment the LLM on several reasoning tasks. This novel latent reasoning paradigm leads to emergent advanced reasoning patterns: the continuous thought can encode multiple alternative next reasoning steps, allowing the model to perform a breadth-first search (BFS) to solve the problem, rather than prematurely committing to a single deterministic path like CoT. COCONUT outperforms CoT in certain logical reasoning tasks that require substantial backtracking during planning, with fewer thinking tokens during inference. These findings demonstrate the promise of latent reasoning and offer valuable insights for future research.

Date: December 12, 2024

Meta

1 Introduction

Large language models (LLMs) have demonstrated remarkable reasoning abilities, emerging from extensive pretraining on human languages (Dubey et al., 2024; Achiam et al., 2023). While next token prediction is an effective training objective, it imposes a fundamental constraint on the LLM as a reasoning machine: the explicit reasoning process of LLMs must be generated in word tokens. For example, a prevalent approach, known as chain-of-thought (CoT) reasoning (Wei et al., 2022), involves prompting or training LLMs to generate solutions step-by-step using natural language. However, this is in stark contrast to certain human cognition results. Neuroimaging studies have consistently shown that the language network – a set of brain regions responsible for language comprehension and production – remains largely inactive during various reasoning tasks (Amalric and Dehaene, 2019; Monti et al., 2012, 2007, 2009; Fedorenko et al., 2011). Further evidence indicates that human language is optimized for communication rather than reasoning (Fedorenko et al., 2024).

A significant issue arises when LLMs use language for reasoning: the amount of reasoning required for each particular reasoning token varies greatly, yet current LLM architectures allocate nearly the same computing budget for predicting every token. Most tokens in a reasoning chain are generated solely for fluency, contributing little to the actual reasoning process. On the contrary, some critical tokens require complex planning and pose huge challenges to LLMs. While previous work has attempted to fix these problems by prompting LLMs to generate succinct reasoning chains (Madaan and Yazdanbakhsh, 2022), or performing additional reasoning before generating some critical tokens (Zelikman et al., 2024), these solutions remain constrained within the language space and do not solve the fundamental problems. On the contrary, it would be ideal for LLMs to have the freedom to reason without any language constraints, and then translate their findings into language only when necessary.

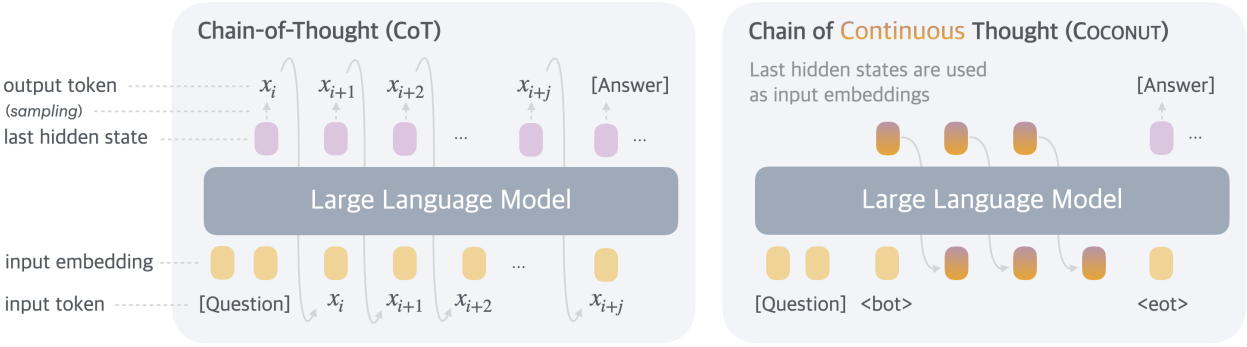

Figure 1 A comparison of Chain of Continuous Thought (Coconut) with Chain-of-Thought (CoT). In CoT, the model generates the reasoning process as a word token sequence (e.g., $[x_i, x_{i+1}, ..., x_{i+j}]$ in the figure). Coconut regards the last hidden state as a representation of the reasoning state (termed "continuous thought"), and directly uses it as the next input embedding. This allows the LLM to reason in an unrestricted latent space instead of a language space.

In this work we instead explore LLM reasoning in a latent space by introducing a novel paradigm, Coconut (Chain of Continuous Thought). It involves a simple modification to the traditional CoT process: instead of mapping between hidden states and language tokens using the language model head and embedding layer, Coconut directly feeds the last hidden state (a continuous thought) as the input embedding for the next token (Figure 1). This modification frees the reasoning from being within the language space, and the system can be optimized end-to-end by gradient descent, as continuous thoughts are fully differentiable. To enhance the training of latent reasoning, we employ a multi-stage training strategy inspired by Deng et al. (2024), which effectively utilizes language reasoning chains to guide the training process.

Interestingly, our proposed paradigm leads to an efficient reasoning pattern. Unlike language-based reasoning, continuous thoughts in Coconut can encode multiple potential next steps simultaneously, allowing for a reasoning process akin to breadth-first search (BFS). While the model may not initially make the correct decision, it can maintain many possible options within the continuous thoughts and progressively eliminate incorrect paths through reasoning, guided by some implicit value functions. This advanced reasoning mechanism surpasses traditional CoT, even though the model is not explicitly trained or instructed to operate in this manner, as seen in previous works (Yao et al., 2023; Hao et al., 2023).

Experimentally, Coconut successfully enhances the reasoning capabilities of LLMs. For math reasoning (GSM8k, Cobbe et al., 2021), using continuous thoughts is shown to be beneficial to reasoning accuracy, mirroring the effects of language reasoning chains. This indicates the potential to scale and solve increasingly challenging problems by chaining more continuous thoughts. On logical reasoning including ProntoQA (Saparov and He, 2022), and our newly proposed ProsQA (Section 4.1) which requires stronger planning ability, Coconut and some of its variants even surpasses language-based CoT methods, while generating significantly fewer tokens during inference. We believe that these findings underscore the potential of latent reasoning and could provide valuable insights for future research.

2 Related Work

Chain-of-thought (CoT) reasoning. We use the term chain-of-thought broadly to refer to methods that generate an intermediate reasoning process in language before outputting the final answer. This includes prompting LLMs (Wei et al., 2022; Khot et al., 2022; Zhou et al., 2022), or training LLMs to generate reasoning chains, either with supervised finetuning (Yue et al., 2023; Yu et al., 2023) or reinforcement learning (Wang et al., 2024; Havrilla et al., 2024; Shao et al., 2024; Yu et al., 2024a). Madaan and Yazdanbakhsh (2022) classified the tokens in CoT into symbols, patterns, and text, and proposed to guide the LLM to generate concise CoT based on analysis of their roles. Recent theoretical analyses have demonstrated the usefulness of CoT from the perspective of model expressivity (Feng et al., 2023; Merrill and Sabharwal, 2023; Li et al., 2024). By employing CoT, the effective depth of the transformer increases because the generated outputs are looped back to the input (Feng et al., 2023). These analyses, combined with the established effectiveness of CoT,

motivated our design that feeds the continuous thoughts back to the LLM as the next input embedding. While CoT has proven effective for certain tasks, its autoregressive generation nature makes it challenging to mimic human reasoning on more complex problems (LeCun, 2022; Hao et al., 2023), which typically require planning and search. There are works that equip LLMs with explicit tree search algorithms (Xie et al., 2023; Yao et al., 2023; Hao et al., 2024), or train the LLM on search dynamics and trajectories (Lehnert et al., 2024; Gandhi et al., 2024; Su et al., 2024). In our analysis, we find that after removing the constraint of a language space, a new reasoning pattern similar to BFS emerges, even though the model is not explicitly trained in this way.

Latent reasoning in LLMs. Previous works mostly define latent reasoning in LLMs as the hidden computation in transformers (Yang et al., 2024; Biran et al., 2024). Yang et al. (2024) constructed a dataset of two-hop reasoning problems and discovered that it is possible to recover the intermediate variable from the hidden representations. Biran et al. (2024) further proposed to intervene the latent reasoning by "back-patching" the hidden representation. Shalev et al. (2024) discovered parallel latent reasoning paths in LLMs. Another line of work has discovered that, even if the model generates a CoT to reason, the model may actually utilize a different latent reasoning process. This phenomenon is known as the unfaithfulness of CoT reasoning (Wang et al., 2022; Turpin et al., 2024). To enhance the latent reasoning of LLM, previous research proposed to augment it with additional tokens. Goyal et al. (2023) pretrained the model by randomly inserting a learnable `<pause>` tokens to the training corpus. This improves LLM's performance on a variety of tasks, especially when followed by supervised finetuning with `<pause>` tokens. On the other hand, Pfau et al. (2024) further explored the usage of filler tokens, e.g., "...", and concluded that they work well for highly parallelizable problems. However, Pfau et al. (2024) mentioned these methods do not extend the expressivity of the LLM like CoT; hence, they may not scale to more general and complex reasoning problems. Wang et al. (2023) proposed to predict a planning token as a discrete latent variable before generating the next reasoning step. Recently, it has also been found that one can "internalize" the CoT reasoning into latent reasoning in the transformer with knowledge distillation (Deng et al., 2023) or a special training curriculum which gradually shortens CoT (Deng et al., 2024). Yu et al. (2024b) also proposed to distill a model that can reason latently from data generated with complex reasoning algorithms. These training methods can be combined to our framework, and specifically, we find that breaking down the learning of continuous thoughts into multiple stages, inspired by iCoT (Deng et al., 2024), is very beneficial for the training. Recently, looped transformers (Giannou et al., 2023; Fan et al., 2024) have been proposed to solve algorithmic tasks, which have some similarities to the computing process of continuous thoughts, but we focus on common reasoning tasks and aim at investigating latent reasoning in comparison to language space.

3 Coconut: Chain of Continuous Thought

In this section, we introduce our new paradigm COCONUT (Chain of Continuous Thought) for reasoning in an unconstrained latent space. We begin by introducing the background and notation we use for language models. For an input sequence $x = (x_1, ..., x_T)$, the standard large language model $\mathcal{M}$ can be described as:

$$H_t = \text{Transformer}(E_t)$$
$$\mathcal{M}(x_{t+1} \mid x_{\leq t}) = \text{softmax}(W h_t)$$

where $E_t = [e(x_1), e(x_2), ..., e(x_t)]$ is the sequence of token embeddings up to position t; $H_t \in \mathbb{R}^{t \times d}$ is the matrix of the last hidden states for all tokens up to position t; h_t is the last hidden state of position t, i.e., $h_t = H_t[t, :]$; $e(\cdot)$ is the token embedding function; W is the parameter of the language model head.

Method Overview. In the proposed COCONUT method, the LLM switches between the "language mode" and "latent mode" (Figure 1). In language mode, the model operates as a standard language model, autoregressively generating the next token. In latent mode, it directly utilizes the last hidden state as the next input embedding. This last hidden state represents the current reasoning state, termed as a "continuous thought".

Special tokens `<bot>` and `<eot>` are employed to mark the beginning and end of the latent thought mode, respectively. As an example, we assume latent reasoning occurs between positions i and j, i.e., $x_i =$ `<bot>` and $x_j =$ `<eot>`. When the model is in the latent mode ($i < t < j$), we use the last hidden state from the previous token to replace the input embedding, i.e., $E_t = [e(x_1), e(x_2), ..., e(x_i), h_i, h_{i+1}, ..., h_{t-1}]$.

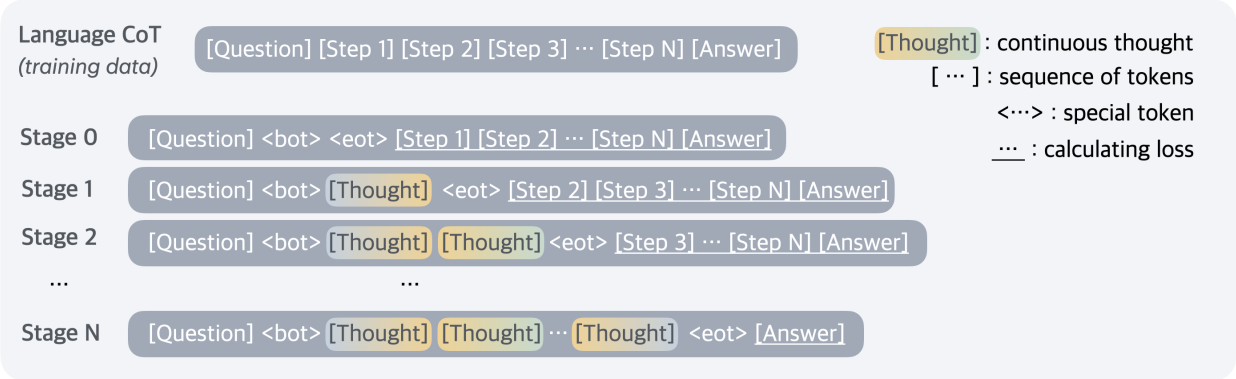

Figure 2 Training procedure of Chain of Continuous Thought (Coconut). Given training data with language reasoning steps, at each training stage we integrate c additional continuous thoughts ($c = 1$ in this example), and remove one language reasoning step. The cross-entropy loss is then used on the remaining tokens after continuous thoughts.

After the latent mode finishes ($t \geq j$), the input reverts to using the token embedding, i.e., $E_t = [e(x_1), e(x_2), ..., e(x_i), h_i, h_{i+1}, ..., h_{j-1}, e(x_j), ..., e(x_t)]$. It is worth noting that the last hidden states have been processed by the final normalization layer, so they are not too large in magnitude. $\mathcal{M}(x_{t+1} \mid x_{\leq t})$ is not defined when $i < t < j$, since the latent thought is not intended to be mapped back to language space. However, softmax(Wh_t) can still be calculated for probing purposes (see Section 4).

Training Procedure. In this work, we focus on a problem-solving setting where the model receives a question as input and is expected to generate an answer through a reasoning process. We leverage language CoT data to supervise continuous thought by implementing a multi-stage training curriculum inspired by Deng et al. (2024). As shown in Figure 2, in the initial stage, the model is trained on regular CoT instances. In the subsequent stages, at the k-th stage, the first k reasoning steps in the CoT are replaced with $k \times c$ continuous thoughts[1], where c is a hyperparameter controlling the number of latent thoughts replacing a single language reasoning step. Following Deng et al. (2024), we also reset the optimizer state when training stages switch. We insert `<bot>` and `<eot>` tokens (which are not counted towards c) to encapsulate the continuous thoughts.

During the training process, we optimize the normal negative log-likelihood loss, but mask the loss on questions and latent thoughts. It is important to note that the objective does **not** encourage the continuous thought to *compress the removed language thought*, but rather to *facilitate the prediction of future reasoning*. Therefore, it's possible for the LLM to learn more effective representations of reasoning steps compared to human language.

Training Details. Our proposed continuous thoughts are fully differentiable and allow for back-propagation. We perform $n + 1$ forward passes when n latent thoughts are scheduled in the current training stage, computing a new latent thought with each pass and finally conducting an additional forward pass to obtain a loss on the remaining text sequence. While we can save any repetitive computing by using a KV cache, the sequential nature of the multiple forward passes poses challenges for parallelism. Further optimizing the training efficiency of Coconut remains an important direction for future research.

Inference Process. The inference process for Coconut is analogous to standard language model decoding, except that in latent mode, we directly feed the last hidden state as the next input embedding. A challenge lies in determining when to switch between latent and language modes. As we focus on the problem-solving setting, we insert a `<bot>` token immediately following the question tokens. For `<eot>`, we consider two potential strategies: a) train a binary classifier on latent thoughts to enable the model to autonomously decide when to terminate the latent reasoning, or b) always pad the latent thoughts to a constant length. We found that both approaches work comparably well. Therefore, we use the second option in our experiment for simplicity, unless specified otherwise.

[1] If a language reasoning chain is shorter than k steps, then all the language thoughts will be removed.

4 Experiments

We validate the feasibility of LLM reasoning in a continuous latent space through experiments on three datasets. We mainly evaluate the accuracy by comparing the model-generated answers with the ground truth. The number of newly generated tokens per question is also analyzed, as a measure of reasoning efficiency. We report the clock-time comparison in Appendix B.

4.1 Reasoning Tasks

Math Reasoning. We use GSM8k (Cobbe et al., 2021) as the dataset for math reasoning. It consists of grade school-level math problems. Compared to the other datasets in our experiments, the problems are more diverse and open-domain, closely resembling real-world use cases. Through this task, we explore the potential of latent reasoning in practical applications. To train the model, we use a synthetic dataset generated by Deng et al. (2023).

Logical Reasoning. Logical reasoning involves the proper application of known conditions to prove or disprove a conclusion using logical rules. This requires the model to choose from multiple possible reasoning paths, where the correct decision often relies on exploration and planning ahead. We use 5-hop ProntoQA (Saparov and He, 2022) questions, with fictional concept names. For each problem, a tree-structured ontology is randomly generated and described in natural language as a set of known conditions. The model is asked to judge whether a given statement is correct based on these conditions. This serves as a simplified simulation of more advanced reasoning tasks, such as automated theorem proving (Chen et al., 2023; DeepMind, 2024).

We found that the generation process of ProntoQA could be more challenging, especially since the size of distracting branches in the ontology is always small, reducing the need for complex planning. To fix that, we apply a new dataset construction pipeline using randomly generated DAGs to structure the known conditions. The resulting dataset requires the model to perform substantial planning and searching over the graph to find the correct reasoning chain. We refer to this new dataset as ProsQA (Proof with Search Question-Answering). A visualized example is shown in Figure 6. More details of datasets can be found in Appendix A.

4.2 Experimental Setup

We use a pre-trained GPT-2 (Radford et al., 2019) as the base model for all experiments. The learning rate is set to 1×10^{-4} while the effective batch size is 128. Following Deng et al. (2024), we also reset the optimizer when the training stages switch.

Math Reasoning. By default, we use 2 latent thoughts (i.e., $c = 2$) for each reasoning step. We analyze the correlation between performance and c in Section 4.4. The model goes through 3 stages besides the initial stage. Then, we have an additional stage, where we still use $3 \times c$ continuous thoughts as in the penultimate stage, but remove all the remaining language reasoning chain. This handles the long-tail distribution of reasoning chains longer than 3 steps. We train the model for 6 epochs in the initial stage, and 3 epochs in each remaining stage.

Logical Reasoning. We use one continuous thought for every reasoning step (i.e., $c = 1$). The model goes through 6 training stages in addition to the initial stage, because the maximum number of reasoning steps is 6 in these two datasets. The model then fully reasons with continuous thoughts to solve the problems in the last stage. We train the model for 5 epochs per stage.

For all datasets, after the standard schedule, the model stays in the final training stage, until the 50th epoch. We select the checkpoint based on the accuracy on the validation set. For inference, we manually set the number of continuous thoughts to be consistent with their final training stage. We use greedy decoding for all experiments.

4.3 Baselines and Variants of Coconut

We consider the following baselines: (1) *CoT*: We use the complete reasoning chains to train the language model with supervised finetuning, and during inference, the model generates a reasoning chain before outputting an

Method	GSM8k		ProntoQA		ProsQA	
	Acc. (%)	# Tokens	Acc. (%)	# Tokens	Acc. (%)	# Tokens
CoT	42.9 ±0.2	25.0	98.8 ±0.8	92.5	77.5 ±1.9	49.4
No-CoT	16.5 ±0.5	2.2	93.8 ±0.7	3.0	76.7 ±1.0	8.2
iCoT	30.0*	2.2	99.8 ±0.3	3.0	98.2 ±0.3	8.2
Pause Token	16.4 ±1.8	2.2	77.7 ±21.0	3.0	75.9 ±0.7	8.2
COCONUT (Ours)	34.1 ±1.5	8.2	99.8 ±0.2	9.0	97.0 ±0.3	14.2
- *w/o curriculum*	14.4 ±0.8	8.2	52.4 ±0.4	9.0	76.1 ±0.2	14.2
- *w/o thought*	21.6 ±0.5	2.3	99.9 ±0.1	3.0	95.5 ±1.1	8.2
- *pause as thought*	24.1 ±0.7	2.2	100.0 ±0.1	3.0	96.6 ±0.8	8.2

Table 1 Results on three datasets: GSM8l, ProntoQA and ProsQA. Higher accuracy indicates stronger reasoning ability, while generating fewer tokens indicates better efficiency. *The result is from Deng et al. (2024).

answer. (2) *No-CoT*: The LLM is trained to directly generate the answer without using a reasoning chain. (3) *iCoT* (Deng et al., 2024): The model is trained with language reasoning chains and follows a carefully designed schedule that "internalizes" CoT. As the training goes on, tokens at the beginning of the reasoning chain are gradually removed until only the answer remains. During inference, the model directly predicts the answer. (4) *Pause token* (Goyal et al., 2023): The model is trained using only the question and answer, without a reasoning chain. However, different from *No-CoT*, special `<pause>` tokens are inserted between the question and answer, which are believed to provide the model with additional computational capacity to derive the answer. For a fair comparison, the number of `<pause>` tokens is set the same as continuous thoughts in COCONUT.

We also evaluate some variants of our method: (1) *w/o curriculum*: Instead of the multi-stage training, we directly use the data from the last stage which only includes questions and answers to train COCONUT. The model uses continuous thoughts to solve the whole problem. (2) *w/o thought*: We keep the multi-stage training which removes language reasoning steps gradually, but don't use any continuous latent thoughts. While this is similar to *iCoT* in the high-level idea, the exact training schedule is set to be consistent with COCONUT, instead of *iCoT*. This ensures a more strict comparison. (3) *Pause as thought*: We use special `<pause>` tokens to replace the continuous thoughts, and apply the same multi-stage training curriculum as COCONUT.

4.4 Results and Discussion

We show the overall results on all datasets in Table 1. Continuous thoughts effectively enhance LLM reasoning, as shown by the consistent improvement over *no-CoT*. It even shows better performance than *CoT* on ProntoQA and ProsQA. We describe several key conclusions from the experiments as follows.

"Chaining" continuous thoughts enhances reasoning. In conventional CoT, the output token serves as the next input, which proves to increase the effective depth of LLMs and enhance their expressiveness (Feng et al., 2023). We explore whether latent space reasoning retains this property, as it would suggest that this method could scale to solve increasingly complex problems by chaining multiple latent thoughts.

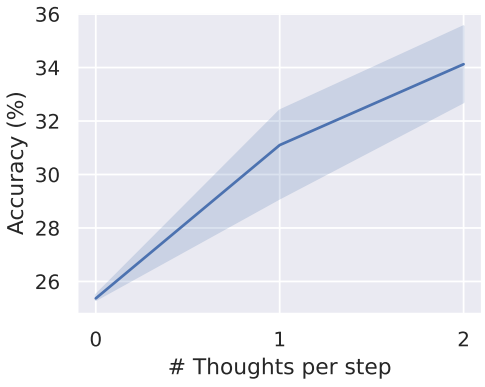

Figure 3 Accuracy on GSM8k with different number of continuous thoughts.

In our experiments with GSM8k, we found that COCONUT outperformed other architectures trained with similar strategies, particularly surpassing the latest baseline, *iCoT* (Deng et al., 2024). The performance is significantly better than COCONUT (*pause as thought*) which also enables more computation in the LLMs. While Pfau et al. (2024) empirically shows that filler tokens, such as the special `<pause>` tokens, can benefit highly parallelizable problems, our results show that COCONUT

architecture is more effective for general problems, e.g., math word problems, where a reasoning step often heavily depends on previous steps. Additionally, we experimented with adjusting the hyperparameter c, which controls the number of latent thoughts corresponding to one language reasoning step (Figure 3). As we increased c from 0 to 1 to 2, the model's performance steadily improved.[2] These results suggest that a chaining effect similar to CoT can be observed in the latent space.

In two other synthetic tasks, we found that the variants of COCONUT (*w/o thoughts* or *pause as thought*), and the *iCoT* baseline also achieve impressive accuracy. This indicates that the model's computational capacity may not be the bottleneck in these tasks. In contrast, GSM8k, being an open-domain question-answering task, likely involves more complex contextual understanding and modeling, placing higher demands on computational capability.

Latent reasoning outperforms language reasoning in planning-intensive tasks. Complex reasoning often requires the model to "look ahead" and evaluate the appropriateness of each step. Among our datasets, GSM8k and ProntoQA are relatively straightforward for next-step prediction, due to intuitive problem structures and limited branching. In contrast, ProsQA's randomly generated DAG structure significantly challenges the model's planning capabilities. As shown in Table 1, *CoT* does not offer notable improvement over *No-CoT*. However, COCONUT, its variants, and *iCoT* substantially enhance reasoning on ProsQA, indicating that latent space reasoning provides a clear advantage in tasks demanding extensive planning. An in-depth analysis of this process is provided in Section 5.

The LLM still needs guidance to learn latent reasoning. In the ideal case, the model should learn the most effective continuous thoughts automatically through gradient descent on questions and answers (i.e., COCONUT *w/o curriculum*). However, from the experimental results, we found the models trained this way do not perform any better than no-CoT.

With the multi-stage curriculum which decomposes the training into easier objectives, COCONUT is able to achieve top performance across various tasks. The multi-stage training also integrates well with pause tokens (COCONUT- *pause as thought*). Despite using the same architecture and similar multi-stage training objectives, we observed a small gap between the performance of *iCoT* and COCONUT (*w/o thoughts*). The finer-grained removal schedule (token by token) and a few other tricks in *iCoT* may ease the training process. We leave combining *iCoT* and COCONUT as future work. While the multi-stage training used for COCONUT has proven effective, further research is definitely needed to develop better and more general strategies for learning reasoning in latent space, especially without the supervision from language reasoning chains.

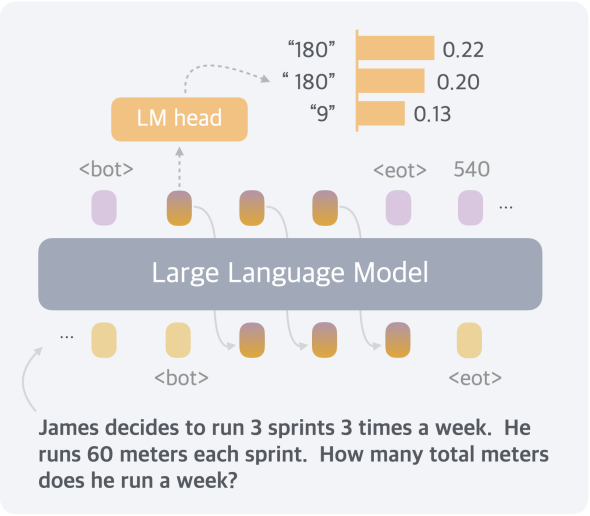

Figure 4 A case study where we decode the continuous thought into language tokens.

Continuous thoughts are efficient representations of reasoning. Though the continuous thoughts are not intended to be decoded to language tokens, we can still use it as an intuitive interpretation of the continuous thought. We show a case study in Figure 4 of a math word problem solved by COCONUT ($c = 1$). The first continuous thought can be decoded into tokens like "180", " 180" (with a space), and "9". Note that, the reasoning trace for this problem should be $3 \times 3 \times 60 = 9 \times 60 = 540$, or $3 \times 3 \times 60 = 3 \times 180 = 540$. The interpretations of the first thought happen to be the first intermediate variables in the calculation. Moreover, it encodes a distribution of different traces into the continuous thoughts. As shown in Section 5.3, this feature enables a more advanced reasoning pattern for planning-intense reasoning tasks.

[2]We discuss the case of larger c in Appendix C.

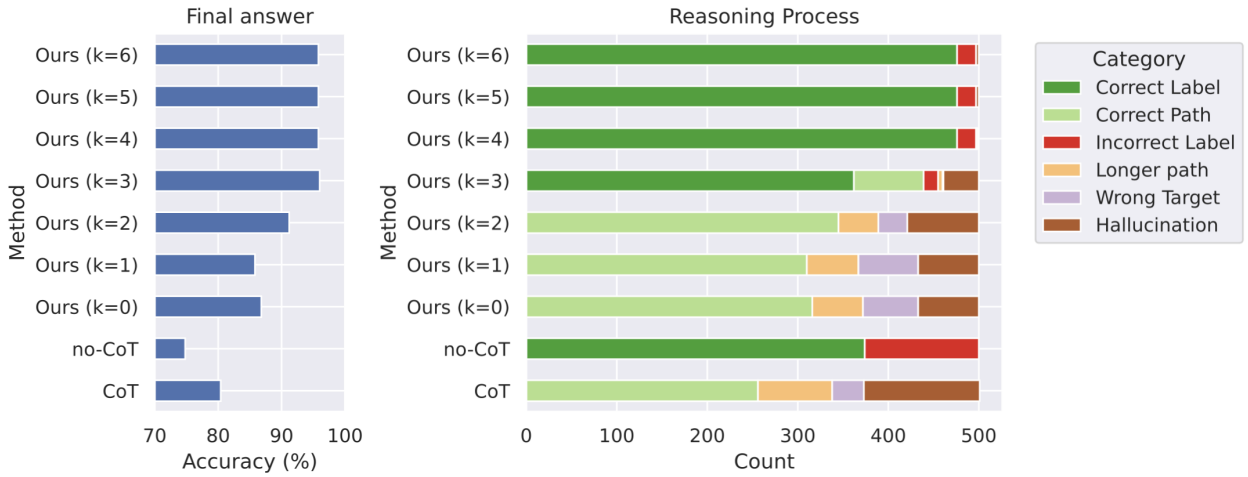

Figure 5 The accuracy of final answer (left) and reasoning process (right) of multiple variants of COCONUT and baselines on ProsQA.

5 Understanding the Latent Reasoning in Coconut

In this section, we present an analysis of the latent reasoning process with a variant of COCONUT. By leveraging its ability to switch between language and latent space reasoning, we are able to control the model to interpolate between fully latent reasoning and fully language reasoning and test their performance (Section 5.2). This also enables us to interpret the the latent reasoning process as tree search (Section 5.3). Based on this perspective, we explain why latent reasoning can make the decision easier for LLMs (Section 5.4).

5.1 Experimental Setup

Methods. The design of COCONUT allows us to control the number of latent thoughts by manually setting the position of the `<eot>` token during inference. When we enforce COCONUT to use k continuous thoughts, the model is expected to output the remaining reasoning chain in language, starting from the $k+1$ step. In our experiments, we test variants of COCONUT on ProsQA with $k \in \{0, 1, 2, 3, 4, 5, 6\}$. Note that all these variants only differ in inference time while sharing the same model weights. Besides, we report the performance of *CoT* and *no-CoT* as references.

To address the issue of forgetting earlier training stages, we modify the original multi-stage training curriculum by always mixing data from other stages with a certain probability ($p = 0.3$). This updated training curriculum yields similar performance and enables effective control over the switch between latent and language reasoning.

Metrics. We apply two sets of evaluation metrics. One of them is based on the correctness of the *final answer*, regardless of the reasoning process. It is the metric used in the main experimental results above (Section 4.4). To enable fine-grained analysis, we define another metric on the *reasoning process*. Assuming we have a complete language reasoning chain which specifies a path in the graph, we can classify it into (1) **Correct Path**: The output is one of the shortest paths to the correct answer. (2) **Longer Path**: A valid path that correctly answers the question but is longer than the shortest path. (3) **Hallucination**: The path includes nonexistent edges or is disconnected. (4) **Wrong Target**: A valid path in the graph, but the destination node is not the one being asked. These four categories naturally apply to the output from COCONUT ($k = 0$) and *CoT*, which generate the full path. For COCONUT with $k > 0$ that outputs only partial paths in language (with the initial steps in continuous reasoning), we classify the reasoning as a Correct Path *if a valid explanation can complete it*. Also, we define Longer Path and Wrong Target for partial paths similarly. If no valid explanation completes the path, it's classified as hallucination. In *no-CoT* and COCONUT with larger k, the model may only output the final answer without any partial path, and it falls into (5) **Correct Label** or (6) **Incorrect Label**. These six categories cover all cases without overlap.

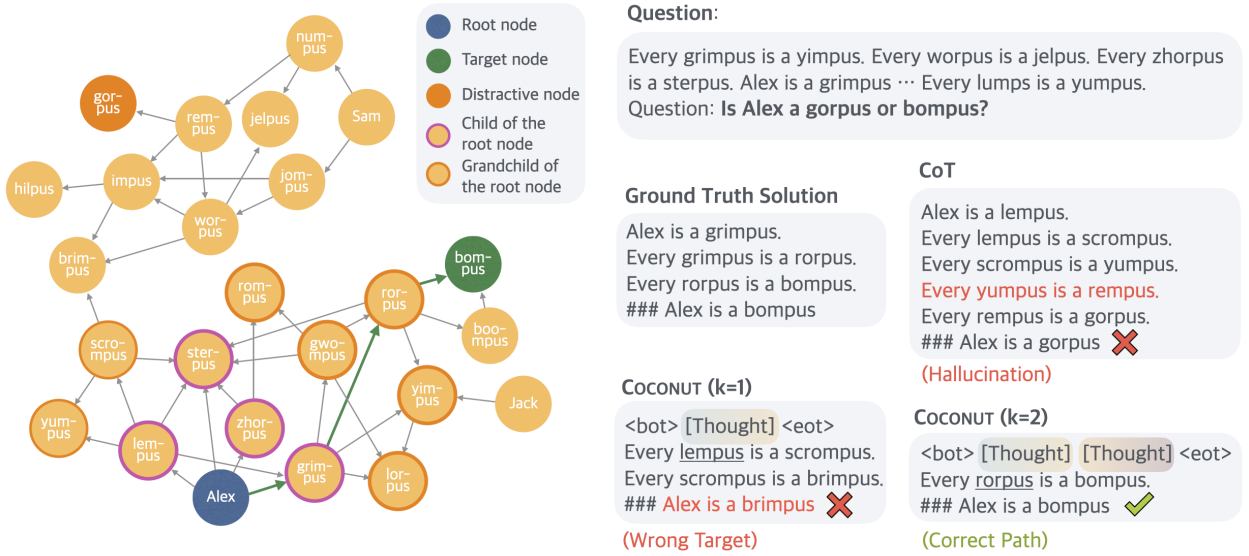

Figure 6 A case study of ProsQA. The model trained with *CoT* hallucinates an edge (*Every yumpus is a rempus*) after getting stuck in a dead end. Coconut (k=1) outputs a path that ends with an irrelevant node. Coconut (k=2) solves the problem correctly.

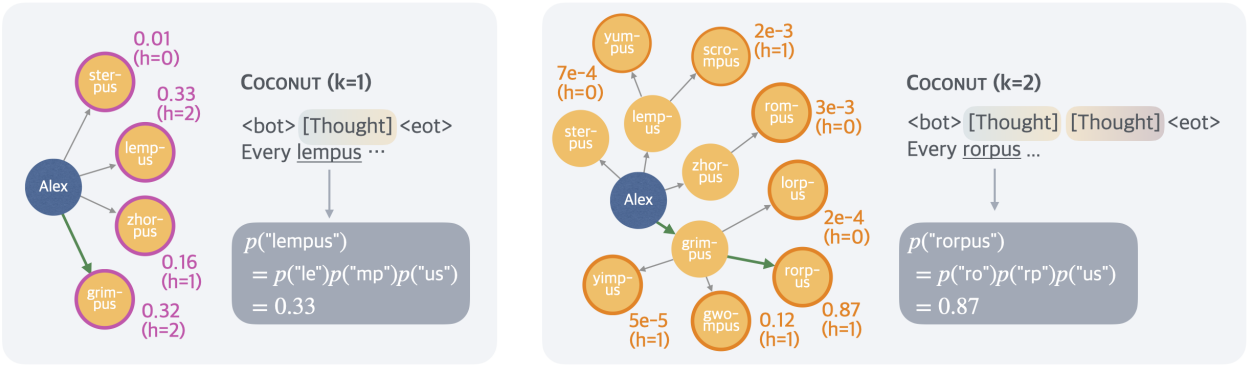

Figure 7 An illustration of the latent search trees. The example is the same test case as in Figure 6. The height of a node (denoted as h in the figure) is defined as the longest distance to any leaf nodes in the graph. We show the probability of the first concept predicted by the model following latent thoughts (e.g., "lempus" in the left figure). It is calculated as the multiplication of the probability of all tokens within the concept conditioned on previous context (omitted in the figure for brevity). This metric can be interpreted as an implicit value function estimated by the model, assessing the potential of each node leading to the correct answer.

5.2 Interpolating between Latent and Language Reasoning

Figure 5 shows a comparative analysis of different reasoning methods on ProsQA. As more reasoning is done with continuous thoughts (increasing k), both final answer accuracy (Figure 5, left) and the rate of correct reasoning processes ("Correct Label" and "Correct Path" in Figure 5, right) improve. Additionally, the rate of "Hallucination" and "Wrong Target" decrease, which typically occur when the model makes a wrong move earlier. This also indicates the better planning ability when more reasoning happens in the latent space.

A case study is shown in Figure 6, where *CoT* hallucinates an nonexistent edge, Coconut ($k = 1$) leads to a wrong target, but Coconut ($k = 2$) successfully solves the problem. In this example, the model cannot accurately determine which edge to choose at the earlier step. However, as latent reasoning can avoid making a hard choice upfront, the model can progressively eliminate incorrect options in subsequent steps and achieves higher accuracy at the end of reasoning. We show more evidence and details of this reasoning process in Section 5.3.

The comparison between *CoT* and Coconut ($k = 0$) reveals another interesting observation: even when

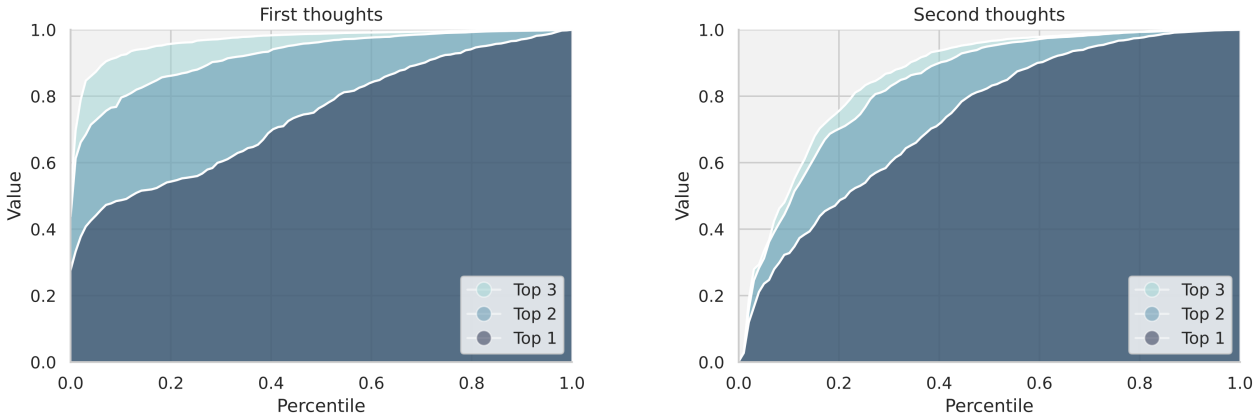

Figure 8 Analysis of parallelism in latent tree search. The left plot depicts the cumulative value of the top-1, top-2, and top-3 candidate nodes for the first thoughts, calculated across test cases and ranked by percentile. The significant gaps between the lines reflect the model's ability to explore alternative latent thoughts in parallel. The right plot shows the corresponding analysis for the second thoughts, where the gaps between lines are narrower, indicating reduced parallelism and increased certainty in reasoning as the search tree develops. This shift highlights the model's transition toward more focused exploration in later stages.

COCONUT is forced to generate a complete reasoning chain, the accuracy of the answers is still higher than *CoT*. The generated reasoning paths are also more accurate with less hallucination. From this, we can infer that the training method of mixing different stages improves the model's ability to plan ahead. The training objective of *CoT* always concentrates on the generation of the immediate next step, making the model "shortsighted". In later stages of COCONUT training, the first few steps are hidden, allowing the model to focus more on future steps. This is related to the findings of Gloeckle et al. (2024), where they propose multi-token prediction as a new pretraining objective to improve the LLM's ability to plan ahead.

5.3 Interpreting the Latent Search Tree

Given the intuition that continuous thoughts can encode multiple potential next steps, the latent reasoning can be interpreted as a search tree, rather than merely a reasoning "chain". Taking the case of Figure 6 as a concrete example, the first step could be selecting one of the children of *Alex*, i.e., *{lempus, sterpus, zhorpus, grimpus}*. We depict all possible branches in the left part of Figure 7. Similarly, in the second step, the frontier nodes will be the grandchildren of *Alex* (Figure 7, right).

Unlike a standard breadth-first search (BFS), which explores all frontier nodes uniformly, the model demonstrates the ability to prioritize promising nodes while pruning less relevant ones. To uncover the model's preferences, we analyze its subsequent outputs in language space. For instance, if the model is forced to switch back to language space after a single latent thought ($k = 1$), it predicts the next step in a structured format, such as "every [Concept A] is a [Concept B]." By examining the probability distribution over potential fillers for [Concept A], we can derive numeric values for the children of the root node *Alex* (Figure 7, left). Similarly, when $k = 2$, the prediction probabilities for all frontier nodes—the grandchildren of *Alex*—are obtained (Figure 7, right).

The probability distribution can be viewed as the model's implicit *value function*, estimating each node's potential to reach the target. As shown in the figure, "lempus", "zhorpus", "grimpus", and "sterpus" have a value of 0.33, 0.16, 0.32, and 0.01, respectively. This indicates that in the first continuous thought, the model has mostly ruled out "sterpus" as an option but remains uncertain about the correct choice among the other three. In the second thought, however, the model has mostly ruled out other options but focused on "rorpus".

Figure 8 presents an analysis of the parallelism in the model's latent reasoning across the first and second thoughts. For the first thoughts (left panel), the cumulative values of the top-1, top-2, and top-3 candidate nodes are computed and plotted against their respective percentiles across the test set. The noticeable gaps between the three lines indicate that the model maintains significant diversity in its reasoning paths at this

stage, suggesting a broad exploration of alternative possibilities. In contrast, the second thoughts (right panel) show a narrowing of these gaps. This trend suggests that the model transitions from parallel exploration to more focused reasoning in the second latent reasoning step, likely as it gains more certainty about the most promising paths.

5.4 Why is a Latent Space Better for Planning?

In this section, we explore why latent reasoning is advantageous for planning, drawing on the search tree perspective and the value function defined earlier. Referring to our illustrative example, a key distinction between *"sterpus"* and the other three options lies in the structure of the search tree: *"sterpus"* is a leaf node (Figure 6). This makes it immediately identifiable as an incorrect choice, as it cannot lead to the target node *"bompus"*. In contrast, the other nodes have more descendants to explore, making their evaluation more challenging.

To quantify a node's exploratory potential, we measure its height in the tree, defined as the shortest distance to any leaf node. Based on this notion, we hypothesize that *nodes with lower heights are easier to evaluate accurately*, as their exploratory potential is limited. Consistent with this hypothesis, in our example, the model exhibits greater uncertainty between *"grimpus"* and *"lempus"*, both of which have a height of 2—higher than the other candidates.

To test this hypothesis more rigorously, we analyze the correlation between the model's prediction probabilities and node heights during the first and second latent reasoning steps across the test set. Figure 9 reveals a clear pattern: the model successfully assigns lower values to incorrect nodes and higher values to correct nodes when their heights are low. However, as node heights increase, this distinction becomes less pronounced, indicating greater difficulty in accurate evaluation.

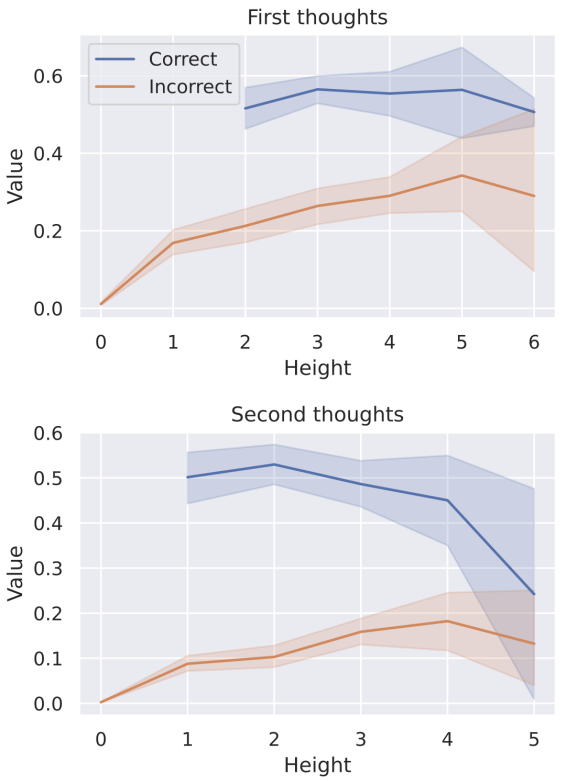

Figure 9 The correlation between prediction probability of concepts and their heights.

In conclusion, these findings highlight the benefits of leveraging latent space for planning. By delaying definite decisions and expanding the latent reasoning process, the model pushes its exploration closer to the search tree's terminal states, making it easier to distinguish correct nodes from incorrect ones.

6 Conclusion

In this paper, we presented COCONUT, a novel paradigm for reasoning in continuous latent space. Through extensive experiments, we demonstrated that COCONUT significantly enhances LLM reasoning capabilities. Notably, our detailed analysis highlighted how an unconstrained latent space allows the model to develop an effective reasoning pattern similar to BFS. Future work is needed to further refine and scale latent reasoning methods. One promising direction is pretraining LLMs with continuous thoughts, which may enable models to generalize more effectively across a wider range of reasoning scenarios. We anticipate that our findings will inspire further research into latent reasoning methods, ultimately contributing to the development of more advanced machine reasoning systems.

SoftCoT: Soft Chain-of-Thought for Efficient Reasoning with LLMs

Yige Xu[1,2]*, **Xu Guo**[1,*†], **Zhiwei Zeng**[1†], **Chunyan Miao**[1,2]
[1]Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly
[2]College of Computing and Data Science
Nanyang Technological University, Singapore
{yige002,xu008}@e.ntu.edu.sg, {zhiwei.zeng,ascymiao}@ntu.edu.sg

Abstract

Chain-of-Thought (CoT) reasoning enables Large Language Models (LLMs) to solve complex reasoning tasks by generating intermediate reasoning steps. However, most existing approaches focus on hard token decoding, which constrains reasoning within the discrete vocabulary space and may not always be optimal. While recent efforts explore continuous-space reasoning, they often require full-model fine-tuning and suffer from catastrophic forgetting, limiting their applicability to state-of-the-art LLMs that already perform well in zero-shot settings with a proper instruction. To address this challenge, we propose a novel approach for continuous-space reasoning that does not require modifying the LLM. Specifically, we employ a lightweight fixed assistant model to generate instance-specific soft thought tokens speculatively as the initial chain of thoughts, which are then mapped into the LLM's representation space via a trainable projection module. Experimental results on five reasoning benchmarks demonstrate that our method enhances LLM reasoning performance through supervised, parameter-efficient fine-tuning.

1 Introduction

In recent years, Large Language Models (LLMs) have become a cornerstone in Natural Language Processing (NLP), exhibiting advanced natural language understanding and generation (Brown et al., 2020; Chowdhery et al., 2023; OpenAI, 2023; Dubey et al., 2024; Yang et al., 2024). Scaling model sizes has not only improved instruction-following (Kojima et al., 2022) but also triggered emergent reasoning abilities, as first evidenced by chain-of-thought (CoT) prompting (Wei et al., 2022). CoT prompts LLMs to generate intermediate reasoning steps before providing the final answer, which not only enhances interpretability but also improves a range of reasoning-intensive tasks (Zhang et al., 2023; Sprague et al., 2024). It has inspired many advanced prompting frameworks, marking a paradigm shift from scaling training-time compute (Kojima et al., 2022) to scaling inference-time compute (Wang et al., 2023; Yao et al., 2023) to further boost LLM performance.

Nevertheless, CoT's effectiveness depends on the quality of intermediate thoughts, as the auto-regressive generation process can propagate errors. To mitigate this challenge, methods like self-consistency (Wang et al., 2023) generate multiple reasoning paths, while Tree-of-Thought (Yao et al., 2023) and Graph-of-Thought (Besta et al., 2024) frameworks organize these paths to select higher-quality steps. Despite these improvements, such methods are computationally inefficient due to the need for extensive thought sampling.

To enhance CoT efficiency, recent research explores skipping the decoding of hard tokens at intermediate steps. Methods like Continuous CoT (Cheng and Durme, 2024) and Coconut (Hao et al., 2024) conduct reasoning in a continuous space by using latent representations instead of discrete token sequences. Their results have shown to be superior to long-sequence discrete reasoning chains using only a short length of continuous representation. Yet, these methods require full-model fine-tuning, which incurs substantial computational costs, risks catastrophic forgetting, and limits their transferability across tasks.

We empirically observe that supervised fine-tuning of the LLaMA3.1-8B (Dubey et al., 2024) model with a language modeling objective on reasoning tasks (which is employed by both Coconut and CCoT) can lead to performance degradation compared with the zero-shot settings. We conjecture that this is due to catastrophic forgetting, a phenomenon also observed by Kalajdzievski (2024) and Lobo et al. (2024). Thus, the methodologies of Coconut, which is based on GPT-2 (Radford et al.,

*The first two authors contributed equally.
†Corresponding authors.

2019), and CCoT, which is built upon LLaMA2-7B (Touvron et al., 2023), may not be directly applicable to more recent models such as LLaMA3.1-8B. Therefore, it is crucial to explore alternative methodologies that mitigate catastrophic forgetting while effectively leveraging continuous reasoning techniques in large-scale, instruction-tuned models, which is the main research question of this work.

To mitigate catastrophic forgetting, a straightforward approach is to freeze the backbone LLM and instead optimize an external model for reasoning. Inspired by prompt tuning (Lester et al., 2021) and speculative decoding (Leviathan et al., 2023), we propose an approach that utilizes an auxiliary small assistant model to generate a sequence of "thought" tokens conditioned on a task instruction followed by a specific instance (Li et al., 2023; Shao et al., 2023). These tokens serve as instance-specific prompts to boost LLM's inference. Such an auxiliary prompting mechanism dynamically adapts to different reasoning tasks, thereby improving generalization while preserving the pre-trained knowledge of the LLM.

To facilitate reasoning in a continuous space, we use soft thought tokens (i.e., the last-layer hidden states from the small assistant model before mapping to the vocabulary space) instead of discrete tokens. This ensures reasoning remains within the continuous latent space. However, a representational gap between the assistant model and the LLM may hinder effective knowledge transfer. To bridge this gap, we train a projection module to map the soft thought tokens generated by the assistant model to the LLM's representation space. Training the projection module for each task can be seen as *soft prompt tuning* for the LLM. The overall **Soft** thoughts for **CoT** (SoftCoT) reasoning framework is illustrated in Figure 1.

To evaluate our proposed SoftCoT, we conduct experiments on five reasoning benchmarks and two state-of-the-art LLM architectures. The five benchmarks we choose include mathematical reasoning, commonsense reasoning, and symbolic reasoning. For further exploration, we create a hard version of the ASDiv dataset (Miao et al., 2020), which requires stronger mathematical reasoning ability. The new dataset is named "ASDiv-Aug" in this paper. Experimental results show that SoftCoT consistently improves accuracy on both public datasets and our augmented ASDiv-Aug dataset, demonstrating the effectiveness of SoftCoT in enhancing LLM's reasoning performance. Moreover, SoftCoT employs parameter-efficient fine-tuning, effectively mitigating catastrophic forgetting seen in full-model fine-tuning. Our results highlight the effectiveness of using an assistant model to generate soft thoughts for enhancing LLMs' reasoning capabilities while preserving their pre-trained knowlegde.

2 Related Works

Early research on chain-of-thought (CoT) reasoning can be traced back to Wei et al. (2022), who first introduced a prompting strategy that guides LLMs through decomposed intermediate reasoning steps using few-shot exemplars. Concurrently, Kojima et al. (2022) demonstrated that LLMs are capable of zero-shot CoT reasoning by simply appending the phrase "Let's think step by step" to the prompt template. This discovery underscored the latent reasoning abilities of LLMs, even in the absence of explicit demonstrations.

Building upon these foundational works, the NLP community has extensively explored the potential of CoT reasoning. As summarized by Chu et al. (2024), recent advancements in CoT methods can be broadly categorized into three areas: (1) *Prompt Construction*, which aims to optimize prompts for improved CoT reasoning (Wei et al., 2022; Kojima et al., 2022; Zhang et al., 2023); (2) *Topological Variants*, which leverage structured representations such as trees and graphs to enhance CoT reasoning (Yao et al., 2023; Besta et al., 2024); and (3) *Enhancement Methods*, which introduce external strategies to further improve CoT reasoning, such as question decomposition (Zhou et al., 2023) and self-consistency decoding (Wang et al., 2023). Despite the effectiveness of these approaches, the majority of existing CoT methods rely on discrete token-by-token generation, which imposes inherent constraints and limits their expressiveness.

To address the limitations of discrete language space, an effective approach is to leverage continuous representation space for reasoning. Coconut (Hao et al., 2024) introduces a Chain-of-Continuous-Thought, while CCoT (Cheng and Durme, 2024) employs Compressed Chain-of-Thought, generating content-rich and continuous contemplation tokens. Heima (Shen et al., 2025) further advances this idea by utilizing a single continuous vector to represent compressed reasoning tokens in multi-modal tasks. However, both Co-

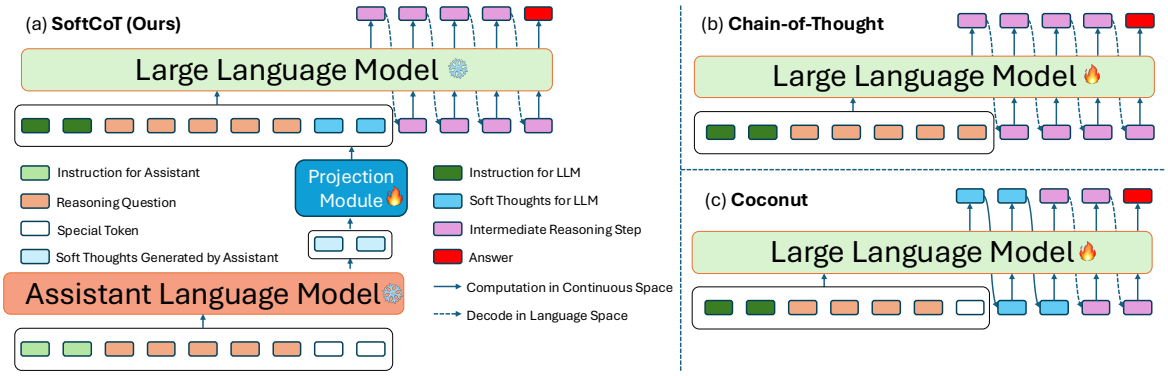

Figure 1: A comparison of SoftCoT, vanilla Chain-of-Thought, and Coconut.

conut and CCoT rely on a language modeling objective for supervised fine-tuning, which is infeasible for state-of-the-art LLMs due to the catastrophic forgetting problem. Moreover, Heima underperforms compared to its backbone model, LLaVA-CoT (Xu et al., 2024). These challenges underscore the need to develop methodologies that mitigate catastrophic forgetting in the application of continuous-space CoT reasoning.

3 Methodology

3.1 Problem Definition and Notations

Given a question $\mathcal{Q} = [q_1, q_2, \cdots, q_{|\mathcal{Q}|}]$, the CoT reasoning will solve the problem on the following two steps: (1) Auto-regressively generate a list of rationale steps $\mathcal{R} = [r_1, r_2, \cdots, r_{|\mathcal{R}|}]$ according to the question; (2) Auto-regressively generate the answer $\mathcal{A} = [a_1, a_2, \cdots, a_{|\mathcal{A}|}]$ according to the question as well as the rationale steps. The generation process can be described as:

$$\begin{aligned} r_{i+1} &= \text{LLM}(\mathcal{Q}; \mathcal{R}_{\leq i}), \\ a_{j+1} &= \text{LLM}(\mathcal{Q}; \mathcal{R}; \mathcal{A}_{\leq j}), \end{aligned} \tag{1}$$

where $\text{LLM}(\cdot)$ indicates a large language model, $\mathcal{R}_{\leq i} = [r_1, \cdots, r_i]$ indicates the previous generated i reasoning tokens, and $\mathcal{A}_{\leq j} = [a_1, \cdots, a_j]$ indicates the previous generated j answer tokens.

The majority of recent works (Zhang et al., 2023; Zhou et al., 2023; Yao et al., 2023) focus on generating discrete hard tokens in $\mathcal{R}$, which is named as "**Hard-CoT**" in this paper. On contrast, some recent works (Hao et al., 2024; Cheng and Durme, 2024) focus on the continuous representations (*a.k.a* latent space) of $\mathcal{R}$, which is named as "**Soft-CoT**" in this paper.

In this paper, we mannually define some rules (e.g., regular expression matching) to extract the framework answer $\hat{\mathcal{A}}$ from the answer $\bar{\mathcal{A}}$ generated by the LLM. Then we compute the accuracy of $\hat{\mathcal{A}}$ comparing with the ground-truth answer $\mathcal{A}$.

3.2 Overview of the SoftCoT Framework

SoftCoT is a novel framework designed to enhance reasoning capabilities in large language models (LLMs). Given an input question $\mathcal{Q}$, the framework produces both a sequence of reasoning steps $\mathcal{R}$ and the final answer $\mathcal{A}$. SoftCoT consists of three key components: the soft thought token generation module, the projection module, and the CoT reasoning module. The overall architecture is illustrated in Figure 1(a).

The soft thought token generation module is inspired by prompt tuning techniques (Lester et al., 2021). In conventional prompt tuning, learnable prompts facilitate the retrieval of knowledge stored within the LLM (Xu et al., 2023). In SoftCoT, soft thought tokens are generated by an assistant language model, which is typically smaller than the backbone LLM (e.g., LLaMA-3.2-1B-Instruct as the assistant model and LLaMA-3.1-8B-Instruct as the backbone model).

A key challenge in this setup is that the assistant model can only generate discrete token sequences as input to the backbone LLM, which imposes constraints and may not always yield optimal prompts. To address this limitation, we introduce continuous soft thought tokens that enable more expressive and flexible prompting. However, a representation gap exists between the assistant model and the backbone LLM, necessitating an intermediate transformation.

To bridge this gap, the projection module maps the soft thought tokens' representations into a space more compatible with the backbone LLM. This ensures that the soft thought tokens effectively guide

the reasoning process.

Finally, the CoT reasoning module leverages both the learned soft thought tokens and word embeddings to generate intermediate reasoning steps $\bar{\mathcal{R}}$ and the final answer $\bar{\mathcal{A}}$. The model is trained using a language modeling objective, optimizing the learnable parameters across the rationale steps and the answer spans.

3.3 Prompt Tuning for CoT Reasoning

Prompt tuning for CoT reasoning aims to optimize the structure and content of the prompt template to enhance the reasoning capabilities of a large language model (LLM). This process can be mathematically formulated as follows:

$$\begin{aligned} \hat{y} &= \mathrm{LLM}\big(P_{\mathbf{p}}(x)\big), \\ \mathbf{p}^* &= \arg\min_{\mathbf{p}} \mathcal{L}(\hat{y}, y), \end{aligned} \tag{2}$$

where $\hat{y}$ represents the predicted output, x denotes the input sequence, and $P_{\mathbf{p}}(x)$ is the input augmented with a prompt $\mathbf{p}$. The objective function $\mathcal{L}(\cdot)$ measures the discrepancy between the model's prediction $\hat{y}$ and the ground-truth label y. The primary goal of prompt tuning is to determine an optimal prompt configuration that effectively guides the LLM to perform CoT reasoning with improved accuracy and interpretability.

A straightforward yet effective approach to optimizing prompts involves leveraging an auxiliary assistant model to generate instance-specific prompts, which provide contextual hints or question summaries to facilitate reasoning (Li et al., 2023; Shao et al., 2023). In this framework, the prompt $\mathbf{p}$ can be decomposed into two components: (1) a fixed, task-specific prompt $\mathbf{p}_{❄}$, which remains constant across all instances and encodes general problem-solving heuristics, and (2) a learnable, instance-specific prompt $\mathbf{p}_{🔥}$, which dynamically adapts to each input instance to provide tailored guidance.

Given the rapid advancements in LLMs, many LLMs are capable of solving complex reasoning tasks under zero-shot settings. Instead of fine-tuning the assistant model for each task, we adopt a more efficient strategy by employing a relatively small, frozen language model to generate $\mathbf{p}_{🔥}$. This approach not only reduces computational costs but also ensures stability and generalization across different problem domains. By systematically integrating instance-specific prompting with fixed task-specific instructions, this method enhances the LLM's reasoning process while maintaining adaptability to various contexts.

3.4 Soft Thought Tokens for CoT Reasoning

One of the advantages of Hard-CoT is that the generated discrete tokens can be tokenized by the LLMs, which does not require a external mapping module. However, there are two main limitations of Hard-CoT: (1) The decoded token space is discrete, which is constrained and sometimes not optimal; (2) The gradient cannot backpropagate to the assistant model since the decoding process cut off the gradient information. A convincing solution is replace the hard tokens to soft thought tokens.

Generating Soft Thought Tokens with an Assistant Model To generate instance-specific soft thoughts, we utilize an auxiliary assistant model that produces soft thoughts based on the given reasoning task. The input to the assistant model, denoted as $\mathbf{x}_{\text{assist}}$, consists of three main components:

$$\mathbf{x}_{\text{assist}} = \big[\mathcal{I}, \mathcal{Q}, \texttt{[UNK]}_{1:N}\big], \tag{3}$$

where

- $\mathcal{I}$ represents the instructional context provided to the assistant model, guiding it on how to generate relevant thoughts.
- $\mathcal{Q}$ denotes the reasoning question that the primary LLM will solve, which has been defined in Section 3.1.
- N `[UNK]` tokens serve as placeholders for the soft thoughts.

Once the input sequence is constructed, the assistant model processes it, and the soft thought tokens are obtained as follows:

$$\begin{aligned} \mathbf{h}^{\text{assist}} &= \mathrm{Assistant}(\mathbf{x}_{\text{assist}}), \\ \mathbf{t}_{\text{assist}} &= \mathbf{h}^{\text{assist}}_{|\mathcal{I}|+|\mathcal{Q}|+1:|\mathcal{I}|+|\mathcal{Q}|+N}. \end{aligned} \tag{4}$$

Here $\mathbf{h}^{\text{assist}}$ denotes the final-layer hidden states of the assistant model, and $\mathbf{t}_{\text{assist}}$ corresponds to the segment of $\mathbf{h}^{\text{assist}}$ associated with the N `[UNK]` tokens. This extracted representation serves as the instance-specific soft thoughts, dynamically adapting to the input reasoning question.

Projection Module Since there exist both a representation gap and a dimensional gap between the assistant language model and the primary LLM, a direct utilization of $\mathbf{t}_{\text{assist}}$ may lead to suboptimal performance. The assistant model and the LLM often operate in different embedding spaces, with distinct hidden state distributions and dimensionalities. To bridge this gap, we introduce a projection module that maps the assistant-generated soft thoughts $\mathbf{t}_{\text{assist}}$ from the assistant model's embedding space to the LLM's embedding space:

$$\mathbf{t}_{\text{🔥}} = \text{Linear}_\theta(\mathbf{t}_{\text{assist}}), \tag{5}$$

where $\text{Linear}_\theta : \mathbb{R}^{d_{\text{assist}}} \rightarrow \mathbb{R}^{d_{\text{LLM}}}$ is a trainable projection layer parameterized by θ. This layer ensures that the assistant-generated soft thoughts are transformed into a suitable format for the LLM, preserving relevant semantic information while adapting to the LLM's feature space.

By incorporating this projection module, we effectively mitigate discrepancies between the assistant model and the LLM, enabling smooth integration of instance-specific thoughts into the CoT reasoning process. This design ensures that the learned thought tokens are both informative and compatible, thereby enhancing the overall reasoning performance of the LLM.

LLM Reasoning with Soft CoT With instance-specific soft thought tokens generated by the assistant model and mapped to the LLM's embedding space, we proceed to the final step: applying these soft thoughts to aid LLMs in CoT reasoning.

The input to the LLM, denoted as $\mathbf{x}_{\text{LLM}}$, follows a structure similar to that of $\mathbf{x}_{\text{assist}}$:

$$\mathbf{x}_{\text{LLM}} = [\mathbf{p}_{\text{❄}}, \mathcal{Q}, \mathbf{t}_{\text{🔥}}], \tag{6}$$

where

- $\mathbf{p}_{\text{❄}}$ is the task-specific instruction, which is a fixed prompt shared across all instances of the same task. It provides general problem-solving heuristics and instructions relevant to the reasoning task.
- $\mathbf{t}_{\text{🔥}}$ is the instance-specific soft thoughts computed by Eq (5). This component dynamically adapts soft thought tokens to each input question, enhancing contextual understanding.

With this structured input, the LLM generates step-by-step reasoning chains, following the principles of CoT reasoning. The reasoning process unfolds by systematically applying logical deductions or problem-solving heuristics, ultimately leading to the generation of the final answer:

$$\begin{aligned} \bar{\mathcal{R}} &= \text{LLM}(\mathbf{x}_{\text{LLM}}), \\ \bar{\mathcal{A}} &= \text{LLM}(\mathbf{x}_{\text{LLM}}, \bar{\mathcal{R}}), \\ \hat{\mathcal{A}} &= \mathcal{E}(\bar{\mathcal{A}}), \end{aligned} \tag{7}$$

where $\mathcal{E}(\cdot)$ is mannual rules for answer extraction.

By integrating both fixed task-specific instructions and instance-specific soft thought tokens, our approach enables the LLM to systematically decompose complex reasoning tasks while leveraging auxiliary knowledge provided by the assistant model. The structured input ensures that the LLM benefits from both general domain knowledge and tailored instance-level guidance, ultimately improving its reasoning effectiveness.

Parameter-Efficient Training In this work, we focus on reasoning tasks that include annotated reasoning steps, which provide explicit intermediate reasoning trajectories leading to the final answer. To effectively train the model, we employ the standard language modeling objective (also known as next-token prediction) to supervise the generation of soft thoughts. During the training stage, the input sequence is structured as follows:

$$\mathbf{x}_{\text{train}} = [\mathbf{p}_{\text{❄}}, \mathcal{Q}, \mathbf{t}_{\text{🔥}}, \mathcal{R}, \mathcal{A}]. \tag{8}$$

To effectively learn the soft thoughts, we apply the negative log-likelihood (NLL) loss over the reasoning steps and the answer span. Specifically, we mask the tokens before the intermediate reasoning steps to prevent the model from directly relying on them during loss computation. Instead, the model is trained to generate the reasoning steps $\mathcal{R}$ and final answer $\mathcal{A}$ in an autoregressive manner.

4 Experiments

4.1 Datasets

We conduct experiments on five reasoning datasets spanning three categories of reasoning: mathematical reasoning, commonsense reasoning, and symbolic reasoning. For mathematical reasoning, we utilize **GSM8K** (Cobbe et al., 2021), **ASDiv** (Miao et al., 2020), and **AQuA** (Ling et al., 2017). For commonsense reasoning, we employ **StrategyQA** (Geva et al., 2021). For symbolic reasoning, we use **Date Understanding** (BIG.Bench.authors, 2023) from the BIG-benchmark.

Given that LLaMA-3.1-8B-Instruct is a well-trained LLM, we augment the ASDiv dataset to ensure that the model encounters novel instances. Specifically, we replicate each instance five times and systematically extract and replace numerical values in the questions with randomly selected alternatives. This augmentation is designed to evaluate the model's reasoning capability rather than its ability to recognize patterns from memorized data. The augmented dataset is named as "**ASDiv-Aug**" in the following part of this paper. All detail statistics of the datasets is shown in Table 1.

4.2 Baselines

We consider the following baselines:

Coconut (Hao et al., 2024). It enables LLMs to reason in a continuous latent space by iteratively feeding hidden states as input embeddings, allowing for more efficient and flexible multi-path reasoning compared to traditional language-based chain-of-thought methods.

Zero-Shot CoT To evaluate whether our trained model has performance degration after supervised fine-tuning, we apply zero-shot CoT based on the prompt templates from Sprague et al. (2024).

Zero-Shot CoT-Unk We directly add some `[UNK]` tokens to represent the un-tuned prompts for CoT reasoning. This baseline exams the effectiveness of the tuned prompts.

Zero-Shot Assist-CoT We directly require the assistant model generates some hard prompt tokens for CoT reasoning. This baseline exams the effectiveness of the tuned soft prompts.

5 Results and Discussions

5.1 Comparison with Baselines

To evaluate SoftCoT, we compare its performance against the baselines introduced in Section 4.2. The results are summarized in Table 2:

(1) **Coconut is not applicable to larger language models**: We modify and run the official implementation of Coconut, adapting it to LLaMA-3.1-8B-Instruct. Our findings indicate that Coconut exhibits performance degradation following supervised fine-tuning with the language modeling objective, which can be attributed to the catastrophic forgetting phenomenon. This observation aligns with findings from prior studies, including Kalajdzievski (2024) and Lobo et al. (2024), which have reported similar issues.

(2) **Incorporating `[UNK]` tokens mitigates performance variance**: We examined the effect of directly adding `[UNK]` tokens as thoughts $\mathbf{t}_{\text{🔥}}$ in Eq. (6). The results demonstrate a slight improvement in overall performance and a reduction in variance. The `[UNK]` token, also known as the "pause token" (Goyal et al., 2024), appears to expand the model's computational capacity, leading to more stable and consistent outputs.

(3) **Assistant model is effective to facilitate CoT reasoning**: We utilize instruction to require the assistant model generate some hard prompts, which can be regarded as the initial thoughts for CoT reasoning. Experiment results show that although it has a larger variance than CoT-Unk, it facilitates the LLM for more diverse CoT generation, which leads to the performance improvement from 68.49 to 69.67 in average.

(4) **SoftCoT consistently benefits from the supervised fine-tuning**: Overall, our proposed SoftCoT consistently outperforms baselines across all five reasoning datasets, involving the mathematical reasoning, the commonsense reasoning, and the symbolic reasoning. The experimental result highlights that our SoftCoT benefits from the supervised fine-tuning and mitigates the catastrophic forgetting problems in state-of-the-art LLMs.

5.2 Generalization to Other LLM Backbones

In addition to LLaMA-3.1, we evaluate SoftCoT on another state-of-the-art LLM family: Qwen2.5 (Yang et al., 2024). Specifically, we select Qwen2.5-7B-Instruct as the backbone LLM to assess the generalization capability of SoftCoT. As shown in Table 3, our analysis yields the following three key findings:

(1) **SoftCoT is effective across different backbone models**: Experimental results on Qwen2.5-7B-Instruct show that SoftCoT consistently improves performance across all reasoning datasets, underscoring its robustness. These findings suggest that SoftCoT serves as a generalizable framework that can be effectively adapted to diverse state-of-the-art LLM architectures.

(2) **SoftCoT enhances LLMs' weaker areas while preserving their strengths**: Experiments on both LLaMA and Qwen LLMs reveal that SoftCoT yields the most significant improvements in

Dataset	Task Type	Answer Type	# Train samples	# Evaluation samples
GSM8K	Mathematical	Number	7,473	1,319
ASDiv-Aug		Number	4,183	1,038
AQuA		Option	97,467	254
StrategyQA	Commonsense	Yes/No	1,832	458
DU	Symbolic	Option	-	250

Table 1: Summary statistics of fiva datasets we used. "-" indicates that there is no training samples available.

Model	GSM8K	ASDiv-Aug	AQuA	StrategyQA	DU	Avg.
	Mathematical			Commonsense	Symbolic	
GPT-2						
Coconut (Hao et al., 2024)	$34.10^{*}_{\pm 1.50}$	$38.92^{\dagger}_{\pm 0.00}$	$22.83^{\dagger}_{\pm 0.00}$	-	-	-
LLaMA-3.1-8B-Instruct						
Zero-Shot CoT	$79.61_{\pm 0.81}$	$86.78_{\pm 0.63}$	$54.65_{\pm 2.43}$	$65.63_{\pm 3.31}$	$54.40_{\pm 2.40}$	68.21
Zero-Shot CoT-Unk	$79.95_{\pm 0.59}$	$86.90_{\pm 0.41}$	$55.28_{\pm 1.88}$	$66.16_{\pm 2.70}$	$54.16_{\pm 1.46}$	68.49
Zero-Shot Assist-CoT	$80.76_{\pm 1.53}$	$86.96_{\pm 0.46}$	$55.83_{\pm 2.98}$	$66.55_{\pm 3.99}$	$58.24_{\pm 3.56}$	69.67
Coconut (Hao et al., 2024)†	$76.12_{\pm 0.00}$	$86.80_{\pm 0.00}$	$53.15_{\pm 0.00}$	-	-	-
SoftCoT (Ours)	$\mathbf{81.03_{\pm 0.42}}$	$\mathbf{87.19_{\pm 0.40}}$	$\mathbf{56.30_{\pm 1.67}}$	$\mathbf{69.04_{\pm 1.23}}$	$\mathbf{59.04_{\pm 1.93}}$	**70.52**

Table 2: Model comparison with baselines. "DU" indicates the Date Understanding (BIG.Bench.authors, 2023) dataset. The first row is under the backbone of GPT-2 (Radford et al., 2019) as backbone. The following rows are under the backbone of LLaMA-3.1-8B-Instruct (Dubey et al., 2024). The last two rows are models trained via the language modeling objective. We run for 5 random seeds and report the average accuracy as well as the standard variance. "*" indicates that the accuracy is reported by Hao et al. (2024). "†" indicates the results that we modify and run the official code of Coconut. ± 0.00 indicates that we only run once for baseline results.

Model	GSM8K	ASDiv-Aug	AQuA	StrategyQA	DU	Avg.
	Mathematical			Commonsense	Symbolic	
Zero-Shot CoT	$83.70_{\pm 0.78}$	$87.19_{\pm 0.28}$	$64.53_{\pm 3.27}$	$49.65_{\pm 3.18}$	$66.40_{\pm 2.26}$	70.29
Zero-Shot CoT-Unk	$84.12_{\pm 0.71}$	$86.94_{\pm 0.89}$	$64.72_{\pm 2.06}$	$50.74_{\pm 1.90}$	$66.48_{\pm 1.43}$	70.60
Zero-Shot Assist-CoT	$84.85_{\pm 0.35}$	$88.63_{\pm 1.05}$	$64.96_{\pm 2.83}$	$52.71_{\pm 2.65}$	$67.04_{\pm 2.84}$	71.64
SoftCoT (Ours)	$\mathbf{85.81_{\pm 1.82}}$	$\mathbf{88.90_{\pm 1.01}}$	$\mathbf{72.44_{\pm 2.19}}$	$\mathbf{60.61_{\pm 1.55}}$	$\mathbf{67.52_{\pm 2.92}}$	**75.06**

Table 3: Model performance using Qwen2.5-7B-Instruct.

commonsense reasoning tasks, where LLMs typically underperform compared to mathematical reasoning. This advantage may stem from SoftCoT's ability to generate contextually relevant continuous thought processes, effectively activating the corresponding knowledge areas within the model. Furthermore, SoftCoT helps mitigate catastrophic forgetting in domains where LLMs already excel, such as mathematical reasoning, thereby preserving and reinforcing existing capabilities.

(3) **SoftCoT facilitates domain transfer**: Given that the Date Understanding dataset lacks training samples, we train the model on other similar datasets and apply zero-shot transfer to evaluate its generalization on Date Understanding. The results indicate that SoftCoT consistently enhances performance in zero-shot domain transfer scenarios, further demonstrating its adaptability.

5.3 Model Analysis and More Studies

5.3.1 Model-Related Factors

To better understand SoftCoT, we conduct experiments to examine the impact of the number of thought tokens. The results, presented in Figure 2,

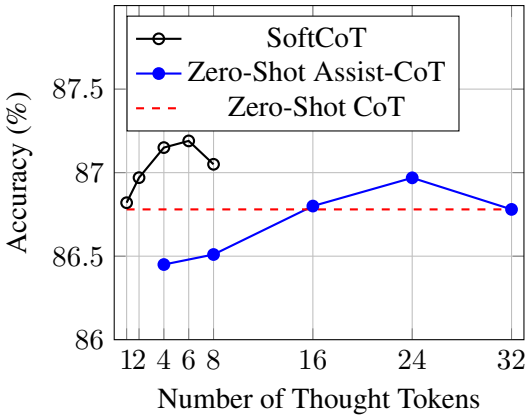

Figure 2: The impact of thought token numbers in ASDiv-Aug using LLaMA-3.1-8B-Instruct.

lead to the following key observations:

(1) **Soft thoughts reduce the required number of thought tokens**: We observe that SoftCoT achieves optimal performance with only six thought tokens, whereas Zero-Shot Assist-CoT requires 24 thought tokens to reach a similar level of effectiveness. This suggests that soft thoughts, which operate in a continuous space, exhibit a stronger representational capacity than hard thoughts expressed in the discrete language space. Our experiments indicate that the optimal number of hard thought tokens is approximately four times that of soft thought tokens, aligning with the 5x ratio reported by Cheng and Durme (2024).

(2) **SoftCoT mitigates the catastrophic forgetting problem**: Experimental results show that SoftCoT consistently outperforms Zero-Shot CoT across all tested numbers of soft thought tokens. In contrast, Zero-Shot Assist-CoT underperforms compared to Zero-Shot CoT when the number of thought tokens is insufficient. This is likely because the assistant model fails to generate a sufficiently informative set of thought tokens under these constraints, introducing noise and leading to confusion in the LLM's reasoning process.

5.3.2 Model-Orthogonal Factors

Self-Consistency (Wang et al., 2023) is a widely adopted technique for enhancing Chain-of-Thought (CoT) reasoning by expanding the search space. One of the most straightforward implementations involves generating multiple CoT reasoning paths and determining the final answer through majority voting. This approach is effective in mitigating errors in reasoning steps by leveraging the diversity of generated thought processes.

Model	$N = 1$	$N = 10$
Zero-Shot CoT	79.61	90.37
Zero-Shot CoT-Unk	79.95	90.43
Zero-Shot Assist-CoT	80.76	90.54
SoftCoT	**81.03**	**90.98**

Table 4: Self Consistency for SoftCoT on LLaMA-3.1-8B-Instruct. "N" indicates the number of reasoning chains.

To further assess the effectiveness of SoftCoT, we conduct experiments incorporating self-consistency. As shown in Table 4, SoftCoT consistently outperforms baseline models, demonstrating that its benefits are complementary to those of self-consistency rather than being redundant or conflicting. This suggests that SoftCoT introduces an independent improvement mechanism, which can be effectively combined with self-consistency for enhanced reasoning performance.

A key advantage of SoftCoT in this context is its ability to provide a more expressive and compact representation of intermediate reasoning steps in continuous space. Unlike traditional CoT reasoning, where discrete thought tokens may introduce inconsistencies or redundant reasoning paths, SoftCoT enables more efficient reasoning trajectories with fewer tokens. This allows self-consistency methods to aggregate results from higher-quality reasoning paths, leading to a more robust and accurate final prediction.

6 Conclusion

In this paper, we introduce SoftCoT, a soft chain-of-thought prompting approach for efficient LLM reasoning. SoftCoT consists of three steps: (1) an assistant model generates soft thought tokens, (2) a projection module trained to map the soft thoughts to LLM's representation space, and (3) the LLM applies soft thoughts for reasoning. To enhance efficiency, SoftCoT speculatively generates *all* the soft thought tokens in a single forward pass. To mitigate the catastrophic forgetting, SoftCoT freezes the backbone LLM and only tunes the projection module. Experiments on five datasets across three types of reason tasks demonstrate the effectiveness of our proposed SoftCoT. Experiments on multiple LLMs as well as orthogonal method such as self-consistency shows the robustness of SoftCoT, which can be adapted in widely scenarios.

Limitations

While SoftCoT represents a promising advancement in Chain-of-Thought (CoT) reasoning within a continuous space, several limitations must be acknowledged.

SoftCoT Cannot Fully Replace the Reasoning Path : Although SoftCoT employs soft thought tokens for reasoning, it does not entirely replace the reasoning path. The decoding stage functions as a search process, which is a crucial component of CoT reasoning. Soft thought tokens primarily serve to enrich the probability space for exploration rather than acting as the search mechanism itself.

Need for Further Empirical Evidence on Scalability : SoftCoT has been evaluated on LLaMA-3.1-8B-Instruct and Qwen2.5-7B-Instruct. However, larger backbone LLMs exist within the same model families. While its success on models with approximately 7–8 billion parameters suggests potential applicability to larger-scale models, its scalability to extremely large LLMs remains an open question and requires thorough empirical validation.

References

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 17682–17690. AAAI Press.

BIG.Bench.authors. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Trans. Mach. Learn. Res.*, 2023.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Jeffrey Cheng and Benjamin Van Durme. 2024. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. PaLM: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. 2024. Navigate through enigmatic labyrinth A survey of chain of thought reasoning: Advances, frontiers and future. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 1173–1203. Association for Computational Linguistics.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? A question answering benchmark with implicit reasoning strategies. *Trans. Assoc. Comput. Linguistics*, 9:346–361.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. 2024. Think before you speak: Training language models with pause tokens. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*.

Damjan Kalajdzievski. 2024. Scaling laws for forgetting when fine-tuning large language models. *arXiv preprint arXiv:2401.05605*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances*