SKIntern: Internalizing Symbolic Knowledge for Distilling Better CoT Capabilities into Small Language Models

Huanxuan Liao[1,2], **Shizhu He**[1,2*], **Yupu Hao**[1,2], **Xiang Li**[1,2],
Yuanzhe Zhang[4], **Jun Zhao**[1,2], **Kang Liu**[1,2,3]
[1] The Key Laboratory of Cognition and Decision Intelligence for Complex Systems,
Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
[3] Shanghai Artificial Intelligence Laboratory, Shanghai, China
[4] National Science Library, Chinese Academy of Sciences, Beijing, China
{liaohuanxuan2023, haoyupu2023, lixiang2022}@ia.ac.cn {shizhu.he, kliu, jzhao}@nlpr.ia.ac.cn

Abstract

Small Language Models (SLMs) are attracting attention due to the high computational demands and privacy concerns of Large Language Models (LLMs). Some studies fine-tune SLMs using Chains of Thought (CoT) data distilled from LLMs, aiming to enhance their reasoning ability. Furthermore, Some CoT distillation methods introduce external symbolic knowledge into the generation process to improve the limited knowledge memory, reasoning ability and out-of-domain (OOD) generalization of SLMs. However, the introduction of symbolic knowledge increases computational overhead and introduces potential noise. In this paper, we introduce *SKIntern*, an innovative approach that empowers SLMs to internalize symbolic knowledge and few-shot examples gradually through a progressive fine-tuning process, guided by a predefined linear decay schedule under curriculum learning. By efficiently internalizing knowledge, *SKIntern* reduces computational overhead and speeds up the reasoning process by focusing solely on the question during inference. It outperforms state-of-the-art baselines by over 5%, while reducing inference costs (measured in FLOPs) by up to $4\times$ across a wide range of SLMs in both in-domain (ID) and out-of-domain (OOD) tasks. Our code will be available at `https://github.com/Xnhyacinth/SKIntern`.

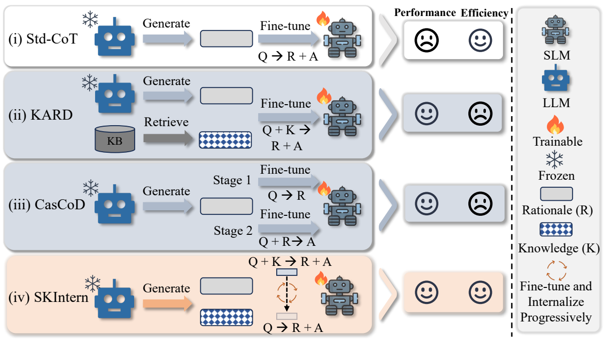

Figure 1: Knowledge utilization comparisons of *SKIntern* and other typical CoT distillation methods. (i) Std-CoT: SLM is fine-tuned to generate the rationale and answer for the question (Q -> R + A). (ii) KARD: Fine-tune the SLM to generate the rationale and answer based on the question and the retrieved symbolic knowledge (Q + K -> R + A). (iii): CasCoD: Decompose the single CoT learning step into two comprehensive learning steps of rationale generation (Q -> R) and rationale utilization (Q + R -> A). (iv): *SKIntern*: Like human interns, SLMs gradually absorb and internalize symbolic knowledge provided by LLMs during the progressive fine-tuning, thereby achieving efficient (Q -> R + A) and effective reasoning (*modeling K in parameters*).

1 Introduction

Large Language Models (LLMs) (Touvron et al., 2023; Yang et al., 2024) have greatly excelled at various complex reasoning tasks such as mathematical (Li et al., 2024a), symbolic (Suzgun et al., 2022) and logical (Dave et al., 2024) reasoning, by applying Chains of Thought (CoT) prompting (Wei et al., 2022) and In-Context Learning (ICL) (Ye et al., 2023; Shum et al., 2023). Nonetheless, the high computational expenses and data privacy issues associated with LLMs have highlighted the need for Small Language Models (SLMs) (Xu et al., 2024). However, these advanced reasoning and knowledge capabilities are typically modeled in larger models ($\geq$13B), making it challenging to replicate in SLMs ($\leq$7B) (Kaplan et al., 2020).

To improve the reasoning ability of SLMs, existing works (Fu et al., 2023; Li et al., 2024b) aim to distill the reasoning ability of LLMs into SLMs by fine-tuning SLMs with high-quality rationales obtained from LLMs, known as standard CoTs distillation (Std-CoT) (Magister et al., 2023). However, due to the limited parameter size of SLMs, they cannot effectively memorize all knowledge and model reasoning ability, making it difficult to generalize to out-of-domain (OOD) tasks.

Recently, several methods have been proposed to

[*]Corresponding author

further improve the knowledge memory and reasoning ability of SLMs. For example, as illustrated in Figure 1, KARD (Kang et al., 2023) uses external knowledge bases to enhance the memory capacity of SLMs, while CasCoD (Dai et al., 2024) employs cascading decomposition to support gradual learning. However, those methods lead to two challenges: 1) **Redundant and noisy symbolic knowledge degrades the effect of CoT distillation**. Document retrieval based on similarity frequently results in repetitive and trivial content, complicating the model's ability to extract key information (Liu et al., 2023). Additionally, retrieved documents often contain irrelevant or misleading information, introducing noise that diminishes the model's performance. 2) **Long input and multi-stage generation reduce the inference efficiency of CoT distillation**. Processing additional documents and rationales imposes significant memory and computational burdens, and the complex inference process complicates deployment and implementation, reducing overall efficiency. Therefore, a key challenge of CoT distillation is: ***Can we effectively and efficiently transfer the rich knowledge and reasoning ability of LLMs through CoT distillation while minimizing computational overhead?***

To resolve the above challenge, we examine the human learning process and draw analogies to model fine-tuning. For instance, at first, an intern typically needs detailed explanations, examples, and documentation to learn new skills (Zou et al., 2024). However, once they have internalized this knowledge and mastered the required skills, such extensive information is no longer needed. Therefore, we believe that if SLMs are provided with detailed guidance and symbolic knowledge while learning rationales from LLMs, their learning outcomes can be greatly enhanced. By gradually internalizing this knowledge into their parameters, SLMs can independently develop efficient reasoning abilities, eliminating the need for additional document retrieval or multi-stage generation.

To perform an efficient and effective CoT distillation, we introduce a novel approach ***SKIntern*** that internalizes the symbolic knowledge during model fine-tuning and enables efficient inference without additional context. Specifically, our method comprises two key steps. Initially, for each training instance, LLMs generate rationales and symbolic knowledge (such as the learning summaries and supplementary materials) and we select the most relevant ones using cosine similarity. Secondly, we gradually perform token-level symbolic knowledge compression and instance-level example pruning based on a predefined linear decay schedule. This refined information is then used to fine-tune the SLM to generate the rationale from the LLMs and the answer. As the schedule progresses, both symbolic knowledge and examples are internalized into the model's parameters, enabling effective reasoning based solely on the questions during inference.

We evaluate *SKIntern* on open-source models like TinyLLaMA (Zhang et al., 2024) and LLaMA2-7B (Touvron et al., 2023) across factual, mathematical, and general reasoning benchmarks. By internalizing symbolic knowledge into parameters and addressing questions exclusively during inference, *SKIntern* surpasses strong baselines in both ID and OOD tasks while significantly reducing computational requirements (measured in FLOPs). This supports our hypothesis that internalizing symbolic knowledge can significantly reduce inference costs, thereby avoiding explicit processing during inference. Additionally, we find that the performance of *SKIntern* can be further enhanced by incorporating few-shot examples into parameters with minimal additional computation. These improvements suggest that our method balances efficiency and effectiveness, making it highly suitable for optimizing SLM inference performance in cost-sensitive scenarios. In conclusion, the contributions of this paper are summarized as follows:

- We propose a novel CoT distillation method *SKIntern* designed to emulate the incremental learning process of interns, gradually learning and mastering knowledge and skills.
- We progressively internalize the symbolic knowledge generated by the LLM and the selected examples into parameters, thereby achieving effective and efficient inference without the need for additional information.
- We conducted extensive experiments on 7 reasoning benchmarks. *SKIntern* outperforms robust baselines by 5% in both ID and OOD tasks, while reducing inference costs by up to $4\times$ across a broad spectrum of SLMs.

2 Related Work

CoT Distillation transfers the reasoning ability of LLMs to SLMs, where reasoning ability is an emergent property that enables LLMs to excel in reasoning tasks through Chains of Thought (CoT) prompting (e.g., Let's think step-by-step) (Wei et al., 2022; Ho et al., 2022). Recent works (Magister et al., 2023; Fu et al., 2023) show that this CoT inference mechanism can be used for distillation: fine-tuning a smaller student model using CoT sequences extracted from a larger teacher model significantly boosts performance. Further studies (Hsieh et al., 2023; Li et al., 2024b) have proposed treating the learning of rationales and answers as distinct optimization objectives. However, these approaches often overlook the limited memory and reasoning ability of SLMs, making it difficult to generalize to OOD tasks. KARD (Kang et al., 2023) boosts SLMs' memory by retrieving external knowledge, while CasCoD (Dai et al., 2024) refines rationale perception through cascading decomposition learning. However, both methods require processing more tokens (document retrieval and multi-stage generation), which introduces additional complexity and uncontrollability in reasoning tasks. Our proposed method mirrors how interns learn a new task by first providing full symbolic knowledge and examples and gradually internalizing them into the parameters, achieving effective inference without additional information.

Prompt Compression condenses lengthy prompts, retaining only essential information while reducing length. This process can be divided into three main methods: Information entropy-based techniques (Li et al., 2023; Jiang et al., 2023) use a small language model to calculate the self-information or error-proneness of tokens, removing those with lower error-proneness; Soft prompts methods (Chevalier et al., 2023; Mu et al., 2023) require fine-tuning LLM parameters to use learnable tokens for condensing prompts; Interpretable summaries methods (Xu et al., 2023; Pan et al., 2024) extract data from the LLM to train models for generating more interpretable text summaries. A method analogous to ours is PromptIntern (Zou et al., 2024), which achieves prompt compression through progressive fine-tuning. We internalize knowledge and examples into the parameters by gradually pruning the prompt during training, allowing the prompt to be discarded during inference.

3 Methodology

In this section, we introduce the detailed procedures of *SKIntern*. As illustrated in Figure 2, *SKIntern* starts with the full knowledge and examples, and progressively prunes tokens to gradually internalize them into the model's parameters, reducing the prompt length and the number of computations towards the model. Below, we first describe how to extract CoT and symbolic knowledge from the teacher LLM in § 3.1. Then we introduce techniques for symbolic knowledge compression and examples pruning to convert them into parameters in § 3.2. Finally, we present a customized progressive fine-tuning pipeline for *SKIntern* in § 3.3. Note, *SKIntern* achieves great results without additional knowledge and examples in input compared with Std-Cot during inference, merely depending on the knowledge stored in the parameters.

3.1 Rationale and Knowledge Generation

Rationale Generation. In our problem setup, we assume a given training dataset $\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{n}$ for the target task, where $\boldsymbol{x}_i$ is the input sequence (question in QA) and $\boldsymbol{y}_i$ is the label (answer in QA). LLMs can generate high-quality rationales, which is known as the emergent ability (Ho et al., 2022). Our objective is to transfer this capability to SLMs through CoT distillation. Firstly, we leverage the CoT prompting (Wei et al., 2022) to guide the teacher LLM in generating proper $\boldsymbol{l}$ rationales for each training data point: $\boldsymbol{r}_{ij} = \text{LLM}(\boldsymbol{p}_c, \boldsymbol{x}_i, \boldsymbol{y}_i)$ where $\boldsymbol{r}$ are generated rationales, $j \in \{1, ..., \boldsymbol{l}\}$ and $\boldsymbol{p}_c$ is the prompt which is shown in Appendix D.1. To maintain high-quality CoT data, we filter out reasoning processes that do not yield correct results, retaining only the distilled CoT sequences that lead to accurate outcomes as the training data (Hsieh et al., 2023).

Symbolic Knowledge Generation. Rationales offer insights into the logic behind answers, which is crucial for SLMs to respond more precisely. However, SLMs with limited parameters may struggle to retain all training data and complex reasoning capabilities, which can affect the quality of rationale generation (Kang et al., 2023). Furthermore, this single learning might lead SLMs to focus on directly answering questions after reading, potentially impairing their ability to generalize in reasoning (Dai et al., 2024). Hence, it is imperative to present the SLM with knowledge in the initial stages of learning to facilitate its understanding.

We use prompt $\boldsymbol{p}_k$ which is in the Appendix D.2 to enable teacher LLM to generate learning summaries $\boldsymbol{k}^m$ that incorporate thinking processes and supplemental knowledge $\boldsymbol{k}^p$, collectively referred to as symbolic knowledge $\boldsymbol{k}$. Formally, the teacher LLM generate $\boldsymbol{m}$ knowledge using the

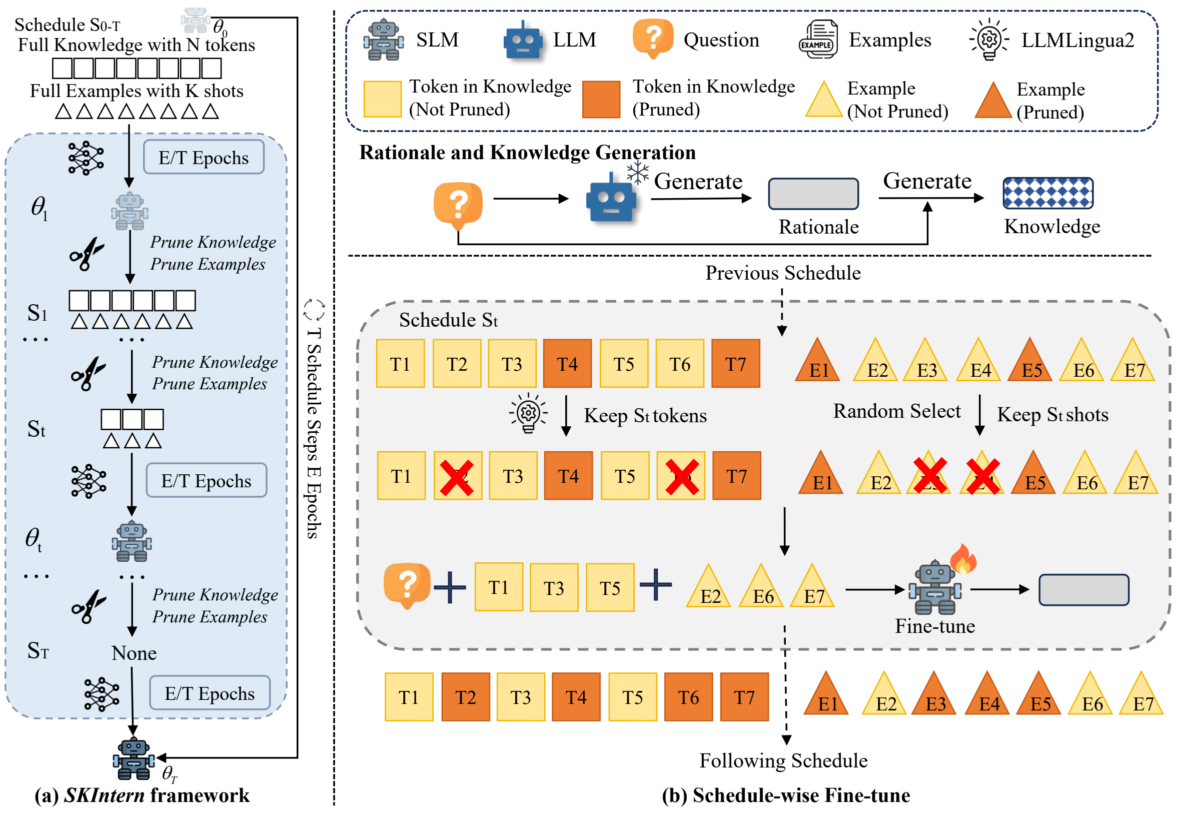

Figure 2: Overview of the *SKIntern* framework. *SKIntern* starts with full symbolic knowledge and examples, and progressively prunes them to gradually internalize knowledge, reducing the prompt length and the number of computations towards the SLM. Based on schedule $\mathcal{S}$, we perform effective knowledge compression and example pruning before fine-tuning the SLM to generate rationales and answers. Gradual fine-tuning makes SLMs internalize knowledge and examples into parameters, thereby enhancing performance without increasing computational cost.

question $\boldsymbol{x}_i$, the rationale $\boldsymbol{r}_i$ and the answer $\boldsymbol{y}_i$: $\boldsymbol{k}_{ij} = \text{LLM}(\boldsymbol{p}_k, \boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{r}_i)$, where $j \in \{1, ..., \boldsymbol{m}\}$. A rationale typically addresses a specific question, whereas knowledge generally offers broader explanations, methods and outlines.

3.2 Progressive Internalization

Before this work, knowledge augmentation has been successfully applied to optimize SLM inference (Kang et al., 2023). However, these methods necessitate processing full knowledge during both training and inference phases, significantly increasing computation overhead. Consequently, they are unsuitable for scenarios with limited computational resources. In contrast, by pruning the number of tokens gradually during the training phase, SKIntern processes only the question during inference without requiring additional symbolic knowledge.

We implement a predefined schedule $\mathcal{S}$ to regulate the pruning rate of knowledge and examples. At each step, the pruned symbolic knowledge and few-shot examples are appended to the question, fine-tuning the SLM over E/T epochs, where the total training spans E epochs. As shown in Figure 2 (a), with T total schedule steps, the value of $\mathcal{S}$ progressively decreases from 1 to 0. As the compression rate increases and fine-tuning progresses, the knowledge in the input gradually reduces to 0, leading to the internalization of knowledge into the model's parameters.

Symbolic Knowledge Compression. Inspired by prompt compression works (Pan et al., 2024), we aim to gradually increase the compression rate to reduce the symbolic knowledge at the token-level determined by $\mathcal{S}_t$ at t-th step and internalize it into the parameters, which can be expressed as:

$$\boldsymbol{k}_i^t = \text{LLMLingua2}(\boldsymbol{k}_i, \mathcal{S}_t) \qquad (1)$$

where LLMLingua2[1] (Pan et al., 2024) is a task-agnostic prompt compression method that distills knowledge from the LLM and fine-tunes the encoder to compress prompts without losing key information, $\boldsymbol{k}^t$ is the compressed symbolic knowledge at t-th step, varying at different schedule $\mathcal{S}_t$.

[1]We apply LLMLingua-2 as the default compressor as it performs the best before June 2024.

Considering that prompt compression is our means of pruning, we directly utilize existing prompt compression methods and models to achieve the compression of knowledge at different learning schedule steps.

Example Pruning. During inference, incorporating few-shot examples can significantly enhance model performance, and incorporating these examples into the fine-tuning stage can further improve the comprehension of various task inputs and outputs (Zou et al., 2024). However, directly adding verbose minority examples to the input would increase the load on the context window and elevate inference computation and latency. So we propose a similarity-based instance-level pruning method to internalize the examples into parameters. For each training instance $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, we begin by employing a relevance scoring function $sim(\cdot, \cdot)$ to assess the similarity between its and different instances in the training set and select the most K relevant examples $\mathcal{D}_i^e$:

$$\mathcal{D}_i^e = \{(\boldsymbol{x}_j, \boldsymbol{y}_j) \mid \boldsymbol{x}_j \in \text{top}\, K(sim(\boldsymbol{x}_i, \boldsymbol{x}_j))\} \quad (2)$$

Inspired by compression techniques, we propose instance-level examples pruning to leverage the performance gains while mitigating the generation of substantial additional overhead. We gradually reduce the number of examples from K to 0 over a total of T schedule steps, to achieve complete example internalization. The number of examples K^t at t-th step can be expressed as:

$$K^t = \lfloor K \times \mathcal{S}_t \rfloor \quad (3)$$

Finally, we randomly select K^t examples from the set $\mathcal{D}_i^e$ as examples $\boldsymbol{e}_i^t$ for t-th step fine-tuning.

3.3 SKIntern Pipeline

Fine-tuning SLMs with Rationales. For each specific schedule step $\mathcal{S}_t$, we utilize the compressed symbolic knowledge $\boldsymbol{k}_i^t$ and pruned examples $\boldsymbol{e}_i^t$ for fine-tuning the SLM p_θ with trainable parameters θ to generate the rationale $\boldsymbol{r}_{ij}$ and answer $\boldsymbol{y}_i$ for the question $\boldsymbol{x}_i$ as follows:

$$\mathcal{L}_t(\theta) = -\frac{1}{n \cdot l} \sum_{i=1}^{n} \sum_{j=1}^{l} \log p_\theta(\boldsymbol{r}_{ij}, \boldsymbol{y}_i \mid \boldsymbol{k}_i^t, \boldsymbol{e}_i^t, \boldsymbol{x}_i) \quad (4)$$

We aim to minimize the negative log-likelihood of the sequence comprising the rationale $\boldsymbol{r}_{ij}$ and answer $\boldsymbol{y}_i$, ensuring rationale precedes the answer.

Progressive Fine-tuning. For a total of T schedule steps, we fine-tune the SLM parameters with the learning rate η for internalizing as follows:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}_t(\theta) \quad (5)$$

Inference. After progressive fine-tuning, we utilize the updated model parameters, denoted as θ_T, to conduct inferences without the need for additional knowledge or examples. Consequently, we can simply handle the question and complete efficient and effective inference.

4 Experiment

In this section, we conduct extensive experiments and comprehensive analysis to evaluate the effectiveness of *SKIntern* on both in-domain (ID) and out-of-domain (OOD) datasets.

4.1 Datasets

Following Ying et al. (2024), we focus on three practical abilities: factual, mathematical, and general reasoning. For each ability, we select a relevant public dataset as the ID dataset, integrate its training data into the target dataset $\mathcal{D}_{\text{train}}$ for mixed training, and combine its test data into the evaluation dataset $\mathcal{D}_{\text{eval}}$. Additionally, each ability includes OOD datasets in $\mathcal{D}_{\text{eval}}$, allowing us to evaluate the model's ability to generalize and enhance performance beyond the ID training environment.

Factual Reasoning: We select the Multitask Language Understanding (MMLU) (Hendrycks et al., 2021a) as the ID dataset, which includes multiple-choice questions across 57 subjects. For OOD evaluation, we use the ARC (Clark et al., 2018), comprising both Easy and Challenge segments.

Mathematical Reasoning: We select MetaMathQA (Yu et al., 2023) as the ID dataset, which only has a training set that includes a high-quality collection of mathematical reasoning question-answer pairs, derived from GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021b). We use GSM8K as the ID evaluation and GSM8K+ (Li et al., 2024a) for OOD evaluation.

General Complex Reasoning: We chose BIG-Bench Hard (BBH) (Suzgun et al., 2022) as the ID dataset, which includes 27 challenging tasks spanning arithmetic, symbolic reasoning, and more, derived from BIG-Bench (BB) (bench authors, 2023). Most of the data consists of multiple-choice questions. For OOD evaluation, we use BB-Sub filtered by CasCoD, and AGIEval (Zhong et al., 2023) subtasks about English multiple-choice questions.

Methods	In-Domain		Out-Of-Domain					Avg	Rel.
	BBH-test	GSM8K	BB-sub	AGIEval	GSM8K-PLUS	ARC-E	ARC-C		FLOPs
# Closed-source model and Open-source models (Zero-shot-CoT)									
GPT-3.5-turbo (*Teacher*)	43.2	72.6	44.0	50.5	55.9	91.8	84.1	63.2	-
LLaMA-3-70B-Instruct	62.6	89.2	51.0	66.3	72.9	97.6	93.2	76.1	-
# TinyLLaMA-1.1B based									
Zero-shot (Radford et al., 2019)	14.0	2.0	17.7	17.8	1.5	19.4	15.0	12.5	×1.0
Zero-shot-CoT (Kojima et al., 2022)	13.5	1.4	17.7	10.4	1.3	16.0	13.4	10.5	×1.0
Fine-tuning	48.8	3.5	26.0	21.2	3.7	28.0	24.6	22.3	×**0.9**
Knowledge-Augmented Fine-tuning	49.3	3.7	27.4	21.9	3.3	29.4	25.3	22.9	×3.7
Std-CoT (Magister et al., 2023)	$47.8_{\pm.43}$	$7.9_{\pm.27}$	$27.6_{\pm.31}$	$21.5_{\pm.56}$	$4.3_{\pm.62}$	$28.2_{\pm.69}$	$25.0_{\pm.48}$	23.2	×1.0
MT-CoT (Li et al., 2024b)	$44.1_{\pm.78}$	$4.1_{\pm.35}$	$25.0_{\pm.45}$	$21.4_{\pm.64}$	$2.8_{\pm.83}$	$33.5_{\pm.52}$	$25.1_{\pm.59}$	22.3	×**0.9**
Step-by-step (Hsieh et al., 2023)	$42.4_{\pm.56}$	$4.3_{\pm.47}$	$26.2_{\pm.38}$	$21.1_{\pm.72}$	$3.1_{\pm.54}$	$29.6_{\pm.61}$	$25.9_{\pm.66}$	21.8	×**0.9**
KARD (BM25) (Kang et al., 2023)	$49.5_{\pm.61}$	$7.6_{\pm.40}$	$26.9_{\pm.43}$	$20.2_{\pm.48}$	$4.0_{\pm.77}$	$28.2_{\pm.85}$	$26.5_{\pm.91}$	23.3	×3.9
CasCoD (Dai et al., 2024)	$48.1_{\pm.49}$	$6.8_{\pm.39}$	$23.1_{\pm.64}$	$19.4_{\pm.73}$	$4.8_{\pm.48}$	$29.0_{\pm.63}$	$27.1_{\pm.42}$	22.6	×3.0
SKIntern (*ours*)	$\mathbf{55.5}_{\pm.71}$	$\mathbf{8.1}_{\pm.65}$	$\mathbf{31.4}_{\pm.44}$	$\mathbf{24.4}_{\pm.90}$	$\mathbf{5.3}_{\pm.68}$	$\mathbf{36.8}_{\pm.89}$	$\mathbf{31.2}_{\pm.32}$	**27.5**	×1.0
# LLaMA2-7B based									
Zero-shot (Radford et al., 2019)	17.3	2.7	18.6	19.2	2.4	25.2	20.6	17.0	×6.4
Zero-shot-CoT (Kojima et al., 2022)	13.5	3.1	12.2	10.3	2.1	29.1	20.2	12.9	×6.4
Fine-tuning	57.8	5.8	33.3	31.0	5.8	73.3	56.3	37.6	×**5.6**
Knowledge-Augmented Fine-tuning	58.7	6.3	34.2	31.8	6.1	75.1	57.0	38.5	×23.7
Std-CoT (Magister et al., 2023)	$58.1_{\pm.74}$	$20.5_{\pm.71}$	$30.7_{\pm.48}$	$23.6_{\pm.65}$	$12.0_{\pm.26}$	$73.4_{\pm.81}$	$55.9_{\pm.78}$	39.2	×6.4
MT-CoT (Li et al., 2024b)	$45.6_{\pm.43}$	$6.8_{\pm.59}$	$27.8_{\pm.75}$	$31.7_{\pm.89}$	$6.0_{\pm.72}$	$74.2_{\pm.46}$	$57.6_{\pm.38}$	35.7	×5.7
Step-by-step (Hsieh et al., 2023)	$54.3_{\pm.37}$	$8.4_{\pm.93}$	$32.9_{\pm.55}$	$32.4_{\pm.64}$	$5.9_{\pm.57}$	$77.7_{\pm.35}$	$61.8_{\pm.87}$	39.1	×**5.6**
KARD (BM25) (Kang et al., 2023)	$58.9_{\pm.53}$	$27.5_{\pm.71}$	$30.3_{\pm.45}$	$18.9_{\pm.38}$	$19.1_{\pm.73}$	$73.7_{\pm.41}$	$57.0_{\pm.82}$	40.8	×24.5
CasCoD (Dai et al., 2024)	$58.9_{\pm.59}$	$29.2_{\pm.75}$	$32.2_{\pm.36}$	$28.8_{\pm.29}$	$\mathbf{21.4}_{\pm.79}$	$74.7_{\pm.91}$	$57.3_{\pm.62}$	43.2	×19.0
SKIntern (*ours*)	$\mathbf{69.3}_{\pm.58}$	$\mathbf{33.9}_{\pm.71}$	$\mathbf{37.2}_{\pm.51}$	$\mathbf{31.3}_{\pm.49}$	$21.2_{\pm.83}$	$\mathbf{78.1}_{\pm.24}$	$\mathbf{62.1}_{\pm.67}$	**47.6**	×6.4

Table 1: Performance (%) of LLaMA2-7B (Touvron et al., 2023) and TinyLLaMA-1.1B (Zhang et al., 2024) with different methods across seven selected datasets. **Bold** indicates the best in each setting. We report the mean and standard deviation of accuracy with 3 different runs for CoT distillation methods. Relative FLOPs cost is calculated relative to the TinyLLaMA with Zero-shot. We calculate the FLOPs required on BBH-test for each method.

4.2 Baselines

We compare our method with the following baselines: *1)* **Teacher & Vanilla Student** in Zero-shot (Radford et al., 2019) and Zero-shot-CoT (Kojima et al., 2022). *2)* **Fine-tuning** involves fine-tuning a model to generate answers given only questions. The performance of the baselines above illustrates the capability of SLMs to solve tasks using only training data, without external guidance or additional knowledge. *3)* **CoT distillation** includes **Std-CoT** (Magister et al., 2023) which is the standard CoT distillation method, enabling direct fine-tuning of the student model with CoT data; **Step-by-step** (Hsieh et al., 2023) is a multi-task method that extracts rationales and answers separately; **MT-CoT** (Li et al., 2024b) is another multi-task method that optimizes both answer prediction and CoT generation simultaneously; **CasCoD** (Dai et al., 2024) decomposes the traditional single-step learning process into two cascaded learning steps. *4)* **Knowledge-Augmentation** involves attaching retrieved passages to the question during both training and inference. This includes **Knowledge-Augmented Fine-tuning** focuses on generating answers only, and **KARD** (Kang et al., 2023) emphasizes learning the generation of rationales.

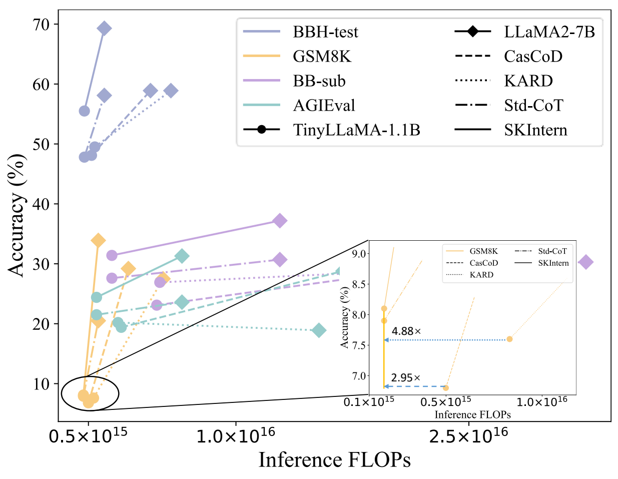

Figure 3: Accuracy (%) against FLOPs for varying model sizes. FLOPs calculations are based on processing all examples from the same task during inference.

4.3 Implementations

For all experiments, we use the LLaMA3-8B, LLaMA2-7B (Touvron et al., 2023), Qwen2 (0.5B, 1.5B, 7B) (Yang et al., 2024) and TinyLLaMA-1.1B (Zhang et al., 2024) as the student SLM. We query the teacher model GPT-3.5-turbo to annotate

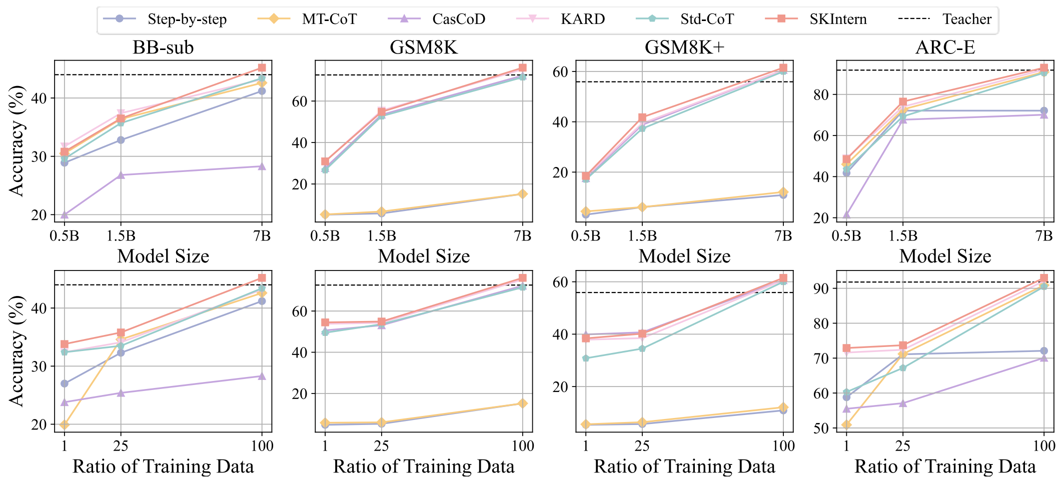

Figure 4: Efficiency on training data and model size. The backbone model for the data size variation is Qwen2-7B.

the CoTs data with the manual prompt (Suzgun et al., 2022). Unless otherwise specified, T is set to 4 (§4.6), and total epochs E is set to 12.

We employ LoRA (Hu et al., 2022) for parameter-efficient fine-tuning of the student SLMs. All experiments are conducted on 2 A100 GPUs with 80GB. During the inference stage, we utilize vLLM (Kwon et al., 2023) to accelerate inference. Detailed information about training, inference and hyperparameters is provided in Appendix A.

4.4 Main Results

We report the performance and inference costs of *SKIntern* and baselines in Table 1 and Figure 3 (More results are shown in Appendix B) and find:

***SKIntern* outperform baselines with fewer FLOPs.** As shown in Figure 3, when FLOPs-matched (in a vertical comparison), *SKIntern* outperforms KARD which retrieves documents to augment reasoning, and CasCoD which enhances reasoning by cascaded decomposition. Specifically, from Table 1, it is evident that *SKIntern* shows an average improvement of 8.4% with LLaMA2-7B and 5.9% with TinyLLaMA-1.1B, respectively. This highlights the utility of dynamic pruning and gradual internalization of symbolic knowledge.

***SKIntern* are up to $4\times$ more efficient than baselines.** Table 1 demonstrates that *SKIntern* uses 2-4$\times$ fewer FLOPs than state-of-the-art KARD and CasCoD. Although other CoT distillation methods can achieve similar computational savings, their performance is significantly worse than *SKIntern* ($\geq$ 8%). Specifically, their performance is 10% lower on the mathematical reasoning dataset GSM8K and 15% lower on the complex reasoning dataset BBH. Furthermore, *SKIntern* achieves comparable performance with fewer FLOPs, as shown in Figure 3 (in a horizontal comparison).

4.5 Efficiency on Dataset and Model Sizes

To evaluate the efficiency of *SKIntern* in terms of training data and model size, we measured test accuracy using Qwen2 (Yang et al., 2024) models across various methods while varying the amount of training data and model size. As shown at the bottom of Figure 4, *SKIntern* successfully transfers the reasoning ability of the teacher LLM into the parameters, even with minimal training data. As the amount of training data increases, *SKIntern* consistently outperforms other baselines, with the improvement magnitude growing as well. This suggests that ***SKIntern* performs optimally across different data volumes and achieves superior reasoning ability distillation**. Even with a limited dataset, *SKIntern* outperforms other methods, demonstrating robustness and sample efficiency.

Regarding model size efficiency, as shown at the top of Figure 4, *SKIntern* outperforms other baselines across various model scales. Notably, *SKIntern* enables Qwen2-7B to surpass the teacher model, GPT-3.5 Turbo, in both ID and OOD tasks, despite having fewer parameters. *SKIntern* offers substantial advantages for models of varying sizes, consistently outperforming other methods. These results underscore the practical benefits of *SKIntern* in resource-limited environments, as it reduces the computational demands for SLMs while delivering performance on par with or surpassing larger models. **This further demonstrates that SLMs (0.5B) struggle to fully leverage CoT reasoning**

SKIntern	BBH	BB	AGIEval	GSM8K+	ARC-E
Pattern of Schedule S					
- exp	64.8	36.2	30.0	16.3	76.0
- $\exp^{-1}$	59.5	31.2	28.8	15.4	73.9
- linear	**69.3**	**37.2**	**31.3**	**21.2**	**78.1**
Step of Schedule T					
- $T = 3$	60.2	33.4	29.1	15.5	74.8
- $T = 4$	**69.3**	**37.2**	**31.3**	**21.2**	**78.1**
- $T = 7$	65.7	35.0	30.0	20.9	76.6

Table 2: Comparison of schedule patterns and steps of *SKIntern*. The backbone model is LLaMA2-7B.

generated by LLMs, highlighting the need for our *SKIntern* approach.

4.6 Analysis on Schedule

Schedule Pattern. We examine the effectiveness of different schedule patterns during the progressive fine-tuning process, focusing on their impact on reasoning performance. The patterns tested include exponential, inverse exponential, and linear decay. As shown in Table 2, the linear decay consistently delivers the highest performance , showcasing superior parsing efficiency and language understanding. In contrast, the inverse exponential schedule exhibits the lowest effectiveness, while the exponential decay offers moderate performance but remains inferior to the linear schedule. These findings indicate that **a gradual, steady reduction is more advantageous than a more aggressive approach**. Progressive fine-tuning with a linear decay schedule appears to yield optimal performance compared to other patterns.

Schedule Setup. We explore the optimal schedule step T for linear decay during progressive fine-tuning. With the total number of epochs set to 12, we chose the common divisors of 12 for linear decay, where T corresponds to the decay step plus 1. As seen in Table 2, $T = 4$ offers the best performance, while $T = 7$ shows slightly lower results, and $T = 3$ yields the poorest performance. This suggests that overly frequent schedule changes hinder sufficient learning in the initial stages, whereas sparse schedules cause large, disruptive jumps, complicating smooth progression and increasing learning difficulty. Therefore, **selecting an appropriate schedule step is crucial for effectively internalizing knowledge and enhancing reasoning abilities in SLMs**.

Methods	BBH	BB	AGIEval	GSM8K+	ARC-E
SKIntern	**69.3**	**37.2**	**31.3**	**21.2**	**78.1**
w/o $\boldsymbol{k}^m$	59.8	30.8	28.7	15.3	74.1
w/o $\boldsymbol{k}^p$	62.3	32.1	29.5	16.2	75.7
w/o $\boldsymbol{e}$	61.9	34.1	29.4	18.1	74.6

Table 3: Ablation studies on different components.

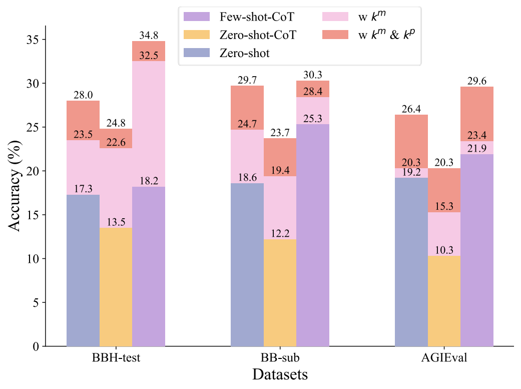

Figure 5: Ablation studies of $\boldsymbol{k}$ on vanilla methods.

4.7 Ablation Studies

To demonstrate the effectiveness of *SKIntern*, we conducted ablation studies using LLaMA2-7B by creating three variants: (1) w/o $\boldsymbol{k}^m$, which removes the learning summary during fine-tuning; (2) w/o $\boldsymbol{k}^p$, where supplemental knowledge is excluded; and (3) w/o $\boldsymbol{e}$, where example pruning is omitted. As shown in Table 3, the removal of any of these components results in reduced performance, highlighting the critical role of internalizing both knowledge and examples in enhancing SLMs' complex reasoning abilities during progressive fine-tuning.

Additionally, we investigate the effectiveness of the generated symbolic knowledge (see Figure 5). Incorporating learning summaries $\boldsymbol{k}^m$ and supplementary knowledge $\boldsymbol{k}^p$ into the original zero-shot, zero-shot-cot, and few-shot-cot significantly enhances performance. Remarkably, this improvement occurs without fine-tuning, demonstrating the utility and generalization of symbolic knowledge in augmenting the model's inference capabilities.

5 Conclusion

In this paper, we introduce *SKIntern*, a novel CoT distillation method designed to internalize symbolic knowledge and rich examples into model parameters, thereby enhancing the ability of SLMs to tackle complex reasoning tasks. Through a systematic schedule, symbolic knowledge generated by the LLM including learning summaries and supplementary knowledge is compressed and selected examples are refined. These elements are then used to fine-tune the SLM, enabling it to produce coherent rationales and accurate answers. We implement a customized progressive fine-tuning pipeline to accommodate various schedule steps and training epochs. Extensive experiments demonstrate that our method not only improves reasoning performance on both in-domain (ID) and out-of-domain (OOD) tasks but also significantly accelerates inference and reduces computational resource usage.

Limitations

Method We have demonstrated through *SKIntern* that the performance of SLM on complex inference tasks can be significantly improved while greatly reducing computational overhead. However, it is important to acknowledge the limitations of our research. The effectiveness of our knowledge enhancement largely depends on the incremental fine-tuning required to internalize the original symbolic knowledge and examples, which increases the complexity and cost of training. Additionally, using LLM to generate supplementary symbolic knowledge necessitates further monetary expenditure due to API calls.

Task While our current tests encompass factual knowledge, mathematics, and complex reasoning, the method's efficacy for different tasks, such as various coding exercises and extended text tasks, requires further analysis and experimentation. Additionally, further investigation is needed to determine which types of symbolic knowledge and task examples are more easily learned and internalized.

Large Language Models Regarding the experiments, given our limited computing and financial budgets, we chose GPT-3.5-Turbo as the teacher. Using GPT-4 would likely better verify the effectiveness of our method, *SKIntern*. Additionally, our aim to enhance the complex reasoning ability of SLMs restricted our choice to mainstream models, such as Llama2, Llama3, and Qwen2, thereby excluding other excellent models like Phi3 and DeepSeek. However, exploring larger LMs such as 13B and 72B with *SKIntern* could be of great interest, presenting a promising direction for future research. Experimental results indicate that enhancing powerful models like Llama3-8B and Qwen2-7B surpasses GPT-3.5-Turbo and matches Llama3-70B.

Ethical Considerations

In this paper, we proposed a novel knowledge enhancement method aimed at leveraging the knowledge of LLMs. However, LLMs may generate inappropriate or discriminatory knowledge. Our approach does not introduce ethical concerns. The datasets we used are public, and there are no privacy issues.

Acknowledgements

This work was supported by the National Key R&D Program of China (No. 2022ZD0160503) and the National Natural Science Foundation of China (No.62376270, No.62276264).

References

BIG bench authors. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3829–3846, Singapore. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457.

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*.

Chengwei Dai, Kun Li, Wei Zhou, and Songlin Hu. 2024. Improve student's reasoning generalizability through cascading decomposed cots distillation. *arXiv preprint arXiv:2405.19842*.

Neisarg Dave, Daniel Kifer, C. Lee Giles, and Ankur Arjun Mali. 2024. Investigating symbolic capabilities of large language models. *ArXiv*, abs/2405.13209.

Yao Fu, Hao-Chun Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. *ArXiv*, abs/2301.12726.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2022. Large language models are reasoning teachers. In *Annual Meeting of the Association for Computational Linguistics*.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *Preprint*, arXiv:2310.06839.

Minki Kang, Seanie Lee, Jinheon Baek, Kenji Kawaguchi, and Sung Ju Hwang. 2023. Knowledge-augmented reasoning distillation for small language models in knowledge-intensive tasks. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, December 10-16, 2023, New Orleans*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Yu Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense passage retrieval for open-domain question answering. *ArXiv*, abs/2004.04906.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024a. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. *ArXiv*, abs/2402.19255.

Shiyang Li, Jianshu Chen, yelong shen, Zhiyu Chen, Xinlu Zhang, Zekun Li, Hong Wang, Jing Qian, Baolin Peng, Yi Mao, Wenhu Chen, and Xifeng Yan. 2024b. Explanations from large language models make small reasoners better. In *2nd Workshop on Sustainable AI*.

Yucheng Li, Bo Dong, Chenghua Lin, and Frank Guerin. 2023. Compressing context to enhance inference efficiency of large language models. *Preprint*, arXiv:2310.06201.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2023. Teaching small language models to reason. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1773–1781, Toronto, Canada. Association for Computational Linguistics.

Jesse Mu, Xiang Lisa Li, and Noah D. Goodman. 2023. Learning to compress prompts with gist tokens. *ArXiv*, abs/2304.08467.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, Dongmei Zhang, Karl Cobbe, Vineet Kosaraju, Mo Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, and Reiichiro Nakano. 2024. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Annual Meeting of the Association for Computational Linguistics*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. *PyTorch: an imperative style, high-performance deep learning library*. Curran Associates Inc., Red Hook, NY, USA.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

KaShun Shum, Shizhe Diao, and Tong Zhang. 2023. Automatic prompt augmentation and selection with chain-of-thought from labeled data. *arXiv preprint arXiv:2302.12822*.

Mirac Suzgun, Nathan Scales, Nathanael Scharli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed Huai hsin Chi, Denny Zhou, and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Annual Meeting of the Association for Computational Linguistics*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. Recomp: Improving retrieval-augmented lms with compression and selective augmentation. *ArXiv*, abs/2310.04408.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. 2024. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Ke-Yang Chen, Kexin Yang, Mei Li, Min Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yunyang Wan, Yunfei Chu, Zeyu Cui, Zhenru Zhang, and Zhi-Wei Fan. 2024. Qwen2 technical report. *ArXiv*.

Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. 2023. Compositional exemplars for in-context learning. In *International Conference on Machine Learning*, pages 39818–39833. PMLR.

Jiahao Ying, Mingbao Lin, Yixin Cao, Wei Tang, Bo Wang, Qianru Sun, Xuanjing Huang, and Shuicheng Yan. 2024. Llms-as-instructors: Learning from errors toward automating model improvement. *arXiv preprint arXiv:2407.00497*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *ArXiv*, abs/2401.02385.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2023. Agieval: A human-centric benchmark for evaluating foundation models. *arXiv preprint arXiv:2304.06364*.

Jiaru Zou, Mengyu Zhou, Tao Li, Shi Han, and Dongmei Zhang. 2024. Promptintern: Saving inference costs by internalizing recurrent prompt during large language model fine-tuning. *arXiv preprint arXiv:2407.02211*.

A Experimantal Settings

A.1 Datasets

For each ability, we select a relevant public dataset, integrate its training data into the target dataset $\mathcal{D}_{\text{train}}$ for mixed training, and combine its test data into the evaluation dataset $\mathcal{D}_{\text{eval}}$. Additionally, each ability includes an OOD dataset in $\mathcal{D}_{\text{eval}}$. This setup allows us to evaluate the model's ability to generalize and enhance performance beyond the ID training environment.

Table 4 shows the statistics details of the selected datasets.

For MMLU (Hendrycks et al., 2021a), we adhere to previous prompt styles (Suzgun et al., 2022), instructing the teacher model (e.g., GPT-3.5-Turbo) to generate answers and Chains of Thought (CoT). By excluding samples with incorrect answers, we ultimately obtained a total of 1,556 samples. For MetaMathQA (Yu et al., 2023), we acquired 3,500 samples through random sampling. For BB (bench authors, 2023), we followed the CasCoD (Dai et al., 2024) methodology by filtering the original dataset for tasks containing the keyword "multiple choice" and randomly extracting up to 100 examples for each task. Note that tasks in BBH do not involve BB-sub.

During the evaluation stage, we use Exact Match (Rajpurkar et al., 2016) as the evaluation metric.

The answer generation between the involved models is conducted in a zero-shot setting, with all models set to a temperature of 0.8 and a maximum token length of 1024. The prompt can be found in the Appendix D.1.

A.2 Hyperparameter

The complete set of stable hyperparameters used for both baseline models and the proposed *SKIntern* training and inference runs can be found in Table 5 and Table 6, respectively.

In our research, we ensured consistent hyperparameter settings across all baselines, including the proposed *SKIntern* method, to maintain the fairness of our comparative analysis. Detailed hyperparameters and their explanations are presented below. For *SKIntern*, particularly in the fourth step, we reduced the enhanced distillation parameters to 3 epochs and fixed the batch size at 8, as the concatenation of specialized knowledge results in longer inputs. We maintained a consistent batch size across all baselines to eliminate any performance differences attributable to varying batch sizes, which depend on model size, with larger models use smaller batch sizes. The learning rate, a key parameter affecting model performance, was set to 5e-5, 1e-4, 2e-4, and 3e-4 in a series of experiments, revealing that larger models require smaller learning rates. Consequently, we adjusted the learning rate according to model size.

A.3 Implementations

Our implementations are based on huggingface transformers v4.42.1 (Wolf et al., 2020) using PyTorch v2.3.1 (Paszke et al., 2019) and LlamaFactory (Zheng et al., 2024).

For CasCoD (Dai et al., 2024), we adhere to the optimal settings recommended by the authors, specifically setting α to 0.3. For KARD (Kang et al., 2023), we employ the BM25 configuration (Robertson and Zaragoza, 2009), a sparse retrieval method based on word frequency, and retrieve three documents per question. Wikipedia serves as the external knowledge base for all datasets. For all retrievers used in *SKIntern*, including BM25, Contriever (Izacard et al., 2021), and DPR (Karpukhin et al., 2020), we utilize the Pyserini[2] library, which offers a reproducible information retrieval framework.

A.4 Symbolic Knowledge Collection

For specialized knowledge collection, using 2-shot hand-written examples, the teacher model is configured with a temperature of 0.8 and a maximum length of 1024 tokens. It generates specialized knowledge corresponding to each incorrect example produced by the student SLMs. The prompt can be found in the Appendix D.2.

B Extended Results

In Table 7, we present the results of various models discussed in this paper, including LLaMA3-8B, QWen2-0.5B, 1.5B, and 7B, utilizing different baseline methods along with the outcomes of *SKIntern*.

C Case Study

We present two cases from Tables 8 and 9 to compare the Chains of Thought (CoTs) generated by *SKIntern*, the teacher large language model (LLM), and the standard CoTs distillation method (Std-CoT). We use ✓ and ✗ to indicate the correctness of the CoT.

Table 8 shows that the Std-CoT's response is confused and fails to comprehend the question accurately. Although it has a rough idea, its rationale is entirely incorrect as it struggles to emulate the rationale of the teacher LLM.

Table 9 presents the symbolic knowledge generated by the LLM for a training example in BBH, encompassing learning summaries and supplementary information. This symbolic knowledge offers

[2]https://github.com/castorini/pyserini

Abilities	**Task**	**# Train**	**# Train (Filtered)**	**# Test**
Factuality	ID: MMLU	Dev + Val: 1,815	1,555	-
	OOD: ARC-C	-	-	1,172
	OOD: ARC-E	-	-	2,376
Math	ID: MetaMathQA	395,000	3,500	-
	OOD: GSM8K	-	-	1,319
	OOD: GSM8K-PLUS	-	-	1,400
Reasoning	ID: BBH	6,511	3,805	1,304
	OOD: BB-sub	-	-	5,384
	OOD: AGIEval	-	-	2,546
All	**Sum**	-	8,860	15,501

Table 4: Statistical details of the selected datasets. Since MMLU lacks official training data, we combined the development and validation datasets to form a training set. To maintain sample balance, we matched the size of MetaMathQA to that of BBH. We obtained balanced samples from two dataset augmentation modes, MATH_Aug and GSM_Aug, resulting in a total of 3,500 samples.

Hyperparameter	**TinyLLaMA-1.1B**	**LLaMA2-7B**	**LLaMA3-8B**	**Qwen2-0.5B**	**Qwen2-1.5B**	**Qwen2-7B**
Max Input Len	2048	4096	4096	4096	4096	4096
Max Output Len	128	128	128	128	128	128
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW
Learning Rate	2e-4	1e-4	5e-5	2e-4	1e-4	1e-4
precision	fp16	fp16	fp16	fp16	fp16	fp16
# Training epochs	12	12	12	12	12	12
# Warmup Steps	10% of total training steps					
Batch Size	32	16	8	32	16	8
Gradient Accumulation	1	2	4	1	2	4
rank of LoRA	32	32	32	32	32	32

Table 5: Training hyperparameters.

Hyperparameter	Student	Teacher	
		Rationale	Reasoning
do_sample	False	True	False
temperature	0.6	0.8	0.6
top-p	0.95	1.0	0.95
top-k	50	50	50
max_new_tokens	1024	2048	1024
# return sequences	1	2	1

Table 6: Generation configs of students and teachers.

detailed logical reasoning and positional insights, which assist the LLM in understanding and solving these problems.

D Instruction Details

D.1 Prompt for Generating CoTs

We use the prompt template below to call the teacher model to generate the CoTs for the training datasets.

Generate CoTs

You are an expert assistant teacher. The following are tasks about {Task_Name}. {Task Description}. Explain your reasoning first and your response should conclude with the format “Therefore, the answer is".

Question: {QUESTION}
Answer: Let’s think step by step.

D.2 Prompt for Specialized Knowledge Collection

Generate Learning Summary only prompts LLMs to analyze the SLM’s errors and generate the specialized knowledge of learning summary.
Generate Learning Summary and Supplementary Knowledge prompts LLMs to analyze the SLM’s errors and generate the specialized knowledge of learning summary and Supplementary Knowledge, providing additional relevant background knowledge to further assist SLMs in solving similar complex reasoning tasks in the future.

Methods	**In-Domain**		**Out-Of-Domain**					**Avg**	**Rel.**
	BBH-test	**GSM8K**	**BB-sub**	**AGIEval**	**GSM8K-PLUS**	**ARC-E**	**ARC-C**		**FLOPs**
# Closed-source model and Open-source models (Zero-shot-CoT)									
GPT-3.5-turbo (*Teacher*)	43.2	72.6	44.0	50.5	55.9	91.8	84.1	63.2	-
LLaMA-3-70B-Instruct	62.6	89.2	51.0	66.3	72.9	97.6	93.2	76.1	-
# LLaMA-3-8B based									
Zero-shot (Radford et al., 2019)	18.2	2.8	27.4	29.7	2.2	50.8	50.0	25.9	×6.2
Zero-shot-CoT (Kojima et al., 2022)	26.5	6.6	23.5	32.2	3.7	68.1	55.5	30.9	×6.2
Fine-tuning	43.7	11.7	29.1	35.3	9.4	75.2	65.2	38.5	×**5.4**
Knowledge-Augmented Fine-tuning	30.4	9.9	14.4	13.0	8.5	40.8	33.9	21.6	×23.3
Std-CoT (Magister et al., 2023)	79.4	61.6	40.5	41.3	45.6	83.2	71.9	60.5	×6.2
MT-CoT (Li et al., 2024b)	62.8	13.1	36.3	**43.9**	11.4	83.6	72.3	46.3	×5.5
Step-by-step (Hsieh et al., 2023)	64.0	11.5	38.8	43.7	9.0	84.3	74.6	46.6	×**5.4**
KARD (BM25) (Kang et al., 2023)	**81.4**	**64.3**	43.1	43.4	**48.6**	85.6	**76.1**	63.2	×24.2
CasCoD (Dai et al., 2024)	32.1	59.1	18.1	23.6	46.1	34.6	27.7	34.5	×17.7
SKIntern (*ours*)	80.8	62.5	42.8	43.6	48.1	**89.9**	75.9	**63.4**	×6.2
# Qwen2-0.5B based									
Std-CoT (Magister et al., 2023)	65.8	26.7	29.6	25.6	17.1	43.6	32.0	34.3	×**0.4**
MT-CoT (Li et al., 2024b)	47.2	5.3	30.5	**27.7**	4.4	46.0	35.1	28.0	×**0.4**
Step-by-step (Hsieh et al., 2023)	44.2	5.2	28.9	26.2	3.1	41.8	36.2	26.5	×**0.4**
KARD (BM25) (Kang et al., 2023)	**66.3**	**30.9**	31.7	23.9	18.2	**48.9**	**37.2**	**36.7**	×1.7
CasCoD (Dai et al., 2024)	37.6	27.7	20.0	15.6	17.6	21.5	14.8	22.1	×1.2
SKIntern (*ours*)	65.9	**30.9**	**30.8**	27.0	**18.5**	48.5	35.6	**36.7**	×**0.4**
# Qwen2-1.5B based									
Std-CoT (Magister et al., 2023)	68.2	52.7	35.7	34.0	37.3	69.3	56.4	50.5	×1.3
MT-CoT (Li et al., 2024b)	58.0	6.7	36.4	34.2	6.1	72.7	57.5	38.8	×**1.1**
Step-by-step (Hsieh et al., 2023)	48.4	5.8	32.8	34.4	6.1	72.1	57.6	36.7	×**1.1**
KARD (BM25) (Kang et al., 2023)	**72.2**	**55.4**	**37.4**	31.2	39.4	74.0	62.2	53.1	×5.2
CasCoD (Dai et al., 2024)	31.7	53.4	25.4	24.7	38.8	57.1	47.8	39.8	×3.8
SKIntern (*ours*)	70.1	54.8	36.5	**36.3**	**41.8**	**76.5**	**62.7**	**54.1**	×1.3
# Qwen2-7B based									
Std-CoT (Magister et al., 2023)	**80.7**	71.5	43.4	**49.9**	60.0	90.5	80.3	68.0	×6.0
MT-CoT (Li et al., 2024b)	70.0	15.2	42.6	49.4	12.1	90.9	80.2	51.5	×5.3
Step-by-step (Hsieh et al., 2023)	68.8	15.2	41.2	49.1	10.9	72.1	71.8	47.0	×**5.2**
KARD (BM25) (Kang et al., 2023)	80.2	75.3	43.2	49.6	60.6	92.1	**83.5**	69.2	×23.6
CasCoD (Dai et al., 2024)	35.7	72.3	23.8	37.4	60.6	70.1	63.1	51.9	×17.4
SKIntern (*ours*)	78.8	**76.1**	**45.2**	49.4	**61.5**	**93.0**	**83.5**	**69.6**	×6.0

Table 7: Performance (%) of LLaMA3-8B (Touvron et al., 2023) and Qwen2-0.5B, 1.5B and 7B (Yang et al., 2024) with different methods across seven selected datasets. **Bold** indicates the best in each setting. Relative FLOPs cost is calculated relative to the TinyLLaMA with Zero-shot. We calculate the FLOPs required on BBH-test for each method.

Question: Kylar went to the store to buy glasses for his new apartment. One glass costs $5, but every second glass costs only 60% of the price. Kylar wants to buy 16 glasses. How much does he need to pay for them?		
	Symbolic Knowledge	**Output**
Teacher	-	The discount price of one glass is 60/100 * 5 = $«60/100*5=3»3. If every second glass is cheaper, that means Kylar is going to buy 16 / 2 = «16/2=8»8 cheaper glasses. So for the cheaper glasses, Kylar is going to pay 8 * 3 = $«8*3=24»24. And for the regular-priced glasses, Kylar will pay 8 * 5 = $«8*5=40»40. So in total Kylar needs to pay 24 + 40 = $«24+40=64»64 for the glasses he wants to buy. Therefore, the answer is 64. ✓
Std-CoT	-	3.0 times a year means Pauline goes to the museum 3.0 * 5.0 = 15.0 times in 5.0 years. The cost of one visit is $2.00, but in the last 5.0 years, it has increased by 1.5 times, so the new cost is $2.00 * 1.5 = $3.00. So, Pauline spent 15.0 * 3.0 = $45.00 on all visits to the museum in dollars. Therefore, the answer is 45. ✗
KARD	1. Conversely, deadweight loss can come from consumers if they buy a product even if it costs more than it benefits them. To describe this, if the same nail market had the government giving a $0.03 subsidy to every nail produced, the subsidy would push the market price of each nail down to $0.07(...) 2. Several glasses are utilized in transparent armor, such as normal plate glass (soda-lime-silica), borosilicate glasses, and fused silica. Plate glass has been the most common glass used due to its low cost(...) 3. To be eligible for the program, applicants must be in financial need, have had a recent eye exam, and have no other resources available to them to pay for glasses, including federal or state programs or assistance from local charitable organizations(...)	The cost of one glass is $5. Kylar wants to buy 16 glasses, so he needs to pay for 16 glasses + 1 glass = 17 glasses. Since every second glass costs 60% of the price, the cost of every second glass is 60% of $5 = $3. So, the total cost of the 17 glasses is $5 x 17 + $3 x (17/2) = $85 + $10. Therefore, Kylar needs to pay $85 + $10 = $95 for the glasses. Therefore, the answer is 95. ✗
SKIntern	-	16 glasses is an even number, so every second glass will cost 60% of the price. Kylar wants to buy 16 glasses, and every second glass costs 60% of the price, so he needs to buy 16/2 = 8 glasses at the full price of $5/glass. He also needs to buy 8 glasses at the reduced price of 60% of $5/glass, which is 60/100 * $5 = $3/glass. So in total, Kylar needs to pay 8 * $5 + 8 * $3 = $40 + $24 = $64 for the glasses. Therefore, the answer is 64. ✓

Table 8: A inference case from GSM8K.

Question: If you follow these instructions, do you return to the starting point?
Take 7 steps. Take 9 steps. Turn around. Turn right.
Options:
- Yes
- No

Symbolic Knowledge

Learning Summary: When solving logical reasoning problems involving directions and movements, it's essential to follow the steps methodically and visualize the movements if necessary. 1. Question Comprehension: Begin by understanding the sequence of movements and turns. Identify the starting point and note the instructions given, such as steps taken and direction changes. 2. Thought Steps: - Start by taking note of the initial position and direction. - Follow the movement instructions step by step: - Take 7 steps in the initial direction. - Take 9 steps in the same direction. - Turn around (180-degree turn) which reverses your direction. - Turn right, which changes your direction perpendicularly. - After executing these steps, assess whether you return to the original position or direction. 3. Visualization: Drawing a simple diagram or using a grid can help track the positions and directions. This visualization helps verify whether the initial and final positions match. 4. Summative Experience: For similar questions, always track each movement and turn carefully. Be aware of the effects of each instruction, particularly turns, which change direction.

Supplementary Knowledge: 1. Understanding Directions: - Familiarize yourself with basic directions (e.g., north, south, east, west) and understand relative turns (left, right, and turn around). - A 180-degree turn changes direction to the opposite, while a 90-degree right or left turn changes the direction perpendicularly. 2. Visualization Techniques: - Use diagrams, sketches, or grids to map directions and movements to see the path clearly. - Visual aids can help prevent confusion, especially when multiple turns are involved. 3. Logical Sequencing: - Carefully follow each step in the sequence as instructed. Misinterpreting a step or turn can lead to incorrect conclusions. - Practice breaking down instructions into smaller parts to manage them more effectively. 4. Definitions: - Turn Around: A 180-degree turn where you face the opposite direction from where you started. - Right Turn: A 90-degree turn to the right, changing the direction perpendicular to the current path. By practicing these steps and understanding the underlying concepts, students can improve their ability to solve similar direction-based logical reasoning problems.

Table 9: A symbolic knowledge generation case from BBH-test.

Generate Learning Summary

As an excellent educational teacher, your goal is to help students enhance their question-solving abilities.
Based on an understanding and explanation of the question, along with relevant background knowledge, fundamental concepts, and empirical conclusions, please generate a learning summary in a numbered list format that will help students complete the same task in the future.

Requirements:
1. Learning summary should outline the thought processes and precautions for addressing student mistakes, including, but not limited to, question comprehension, thought steps and mathematical calculations. It should also provide a summative experience to help students solve similar questions in the future.
2. Ensure that the content is understandable and usable by students, while also being concise and effective.
3. The obtained learning summary should be general and generalized, not aimed at specific questions.
4. Keep these requirements in mind while generating the learning summary and supplementary knowledge.

Return Format:
Return in the following format:
Learning Summary: [Learning Summary]

Question: {QUESTION}
Answer: {ANSWER}
Please follow the requirements and provide the learning summary.

Generate Learning Summary and Supplementary Knowledge

As an excellent educational teacher, your goal is to help students enhance their question-solving abilities and to aid students in completing the same task in the future.
You should generate targeted, detailed thought processes and relevant background knowledge for solving similar questions in the future.
Your role involves creating learning summaries and supplementary knowledge, specifically identifying the steps needed to solve the question and providing additional general knowledge in the supplementary knowledge.

Requirements:
1. Learning summary should outline the thought processes including, but is not limited to, question comprehension, thought steps and mathematical calculations. It should also provide a summative experience to help students solve similar questions in the future.
2. Supplementary knowledge should include a list of essential background information that students need to solve the question. This should encompass, but is not limited to, mathematical formulas, definitions, relevant world knowledge, and specific techniques.
3. Ensure that the content is understandable and usable by students, while also being concise and effective.
4. The obtained learning summary should be general and generalized, not aimed at specific problems, and the supplementary knowledge should also be general knowledge of the problem without involving specific analysis.
5. Keep these requirements in mind while generating the learning summary and supplementary knowledge.

Return Format:
Return in the following format:
Learning Summary: [Learning Summary]
Supplementary Knowledge: [Supplementary Knowledge]

Question: {QUESTION}
Answer: {ANSWER}
Please follow the requirements and provide the learning summary and supplementary knowledge.