SIKeD: Self-guided Iterative Knowledge Distillation for Mathematical Reasoning

Shivam Adarsh[*]
University of Copenhagen
shad@di.ku.dk

Kumar Shridhar[*]
ETH Zurich
shkumar@ethz.ch

Caglar Gulcehre
CLAIRE, EPFL

Nicholas Monath
Google DeepMind

Mrinmaya Sachan
ETH Zurich

Abstract

Large Language Models (LLMs) can transfer their reasoning skills to smaller models by teaching them to generate the intermediate reasoning process required to solve multistep reasoning tasks. While LLMs can accurately solve reasoning tasks through various strategies, even without fine-tuning, smaller models are not expressive enough to fit the LLMs distribution on all strategies when distilled and tend to prioritize one strategy over the others. This reliance on one strategy poses a challenge for smaller models when attempting to solve reasoning tasks that may be difficult with their preferred strategy. To address this, we propose a distillation method SIKeD: **S**elf-guided **I**terative **K**nowledge **D**istillation, where the LLM teaches the smaller model to approach a task using different strategies and the smaller model uses its self-generated on-policy outputs to choose the most suitable strategy for the given task. The training continues in a *self-guided* iterative manner, where for each training iteration, a decision is made on how to combine the LLM data with the self-generated outputs. Unlike traditional distillation methods, SIKeD allows the smaller model to learn *which* strategy is suitable for a given task while continuously learning to solve a task using different strategies. Our experiments on various mathematical reasoning datasets show that SIKeD significantly outperforms traditional distillation techniques across smaller models of different sizes[1].

1 Introduction

Large language models (LLMs), with tens to hundreds of billions of parameters, generally outperform smaller models (with billions of parameters or fewer) in a variety of reasoning tasks (Touvron et al., 2023; Achiam et al., 2023). One notable strength of large models is their ability to reason

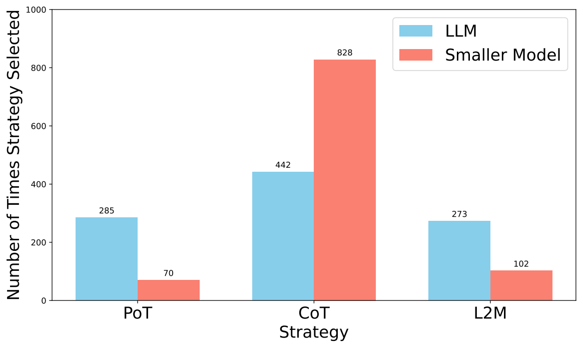

Figure 1: **Histogram of strategy choices for the LLM and the smaller model.** LLM tends to select several reasoning strategies, but the smaller model is biased towards one strategy. The comparison was done on 1K randomly sampled datapoints in the GSM8K train set.

and perform multistep reasoning tasks, often considered an important aspect of intelligence (Gómez-Veiga et al., 2018). However, the significant size and computational demands of these large models present several challenges. For example, LLaMA3 models (Touvron et al., 2023) are trained using clusters of 24,000 GPUs, limiting their accessibility to many researchers and practitioners.

To bridge this gap, a key approach involves teaching smaller models to replicate the knowledge of a larger model, often referred to as *knowledge distillation* (Hinton et al., 2015). Typically, smaller models can be taught to replicate the multistep reasoning capabilities of larger models by incorporating a set of intermediate sequences (Kim and Rush, 2016; Shridhar et al., 2023). However, these intermediate steps can be derived from several different strategies, such as Chain of Thought (CoT) (Wei et al., 2022), Subquestion Decomposition (Shridhar et al., 2022; Zhou et al., 2023), and Program of Thoughts (PoT) (Chen et al., 2023), among others. A viable solution is to distill these reasoning capabilities into smaller models either by distilling individual strategies (Magister et al., 2023; Shridhar et al., 2023; Hsieh et al., 2023) or by incorporating

[*]Equal contribution
[1]Our code is available on Github

multiple strategies simultaneously (Chenglin et al., 2023; Zhu et al., 2024). Although smaller models have demonstrated impressive performance when distilled with a single strategy, they often struggle to master multiple strategies equally well. An example is presented in Figure 1 where a larger model (LLaMA-3 70B) can use multiple strategies to generate data but upon distilling, a smaller model (Gemma 2B) tends to favor one over the others. We used three-shot prompting on both the larger and smaller model, showing the models one example of each strategy. The goal for the model was then to 1) select a strategy and 2) solve the reasoning task using the selected strategy for a given problem. The results displayed in Figure 1 confirm our hypothesis. This happens because reasoning through a variety of strategies tends to emerge as a result of scaling language models, making it difficult for smaller models to replicate this behavior (Lyu et al., 2024).

On the other hand, learning to solve a task using multiple strategies can help smaller models overcome the limitations of relying on a single approach. However, a key challenge arises when, despite being trained on a fixed dataset containing various strategies, a distribution mismatch occurs between the data generated by the LLM and the outputs produced by the smaller model during inference. This mismatch can hinder the ability of the smaller model to generalize across different reasoning strategies. This issue, often discussed in imitation learning (Pomerleau, 1991; Ross and Bagnell, 2010), results in the student model consistently choosing one strategy, even when a different approach would be more appropriate. As a result, the student generates outputs with strategy choices that are highly unlikely to match those produced by the teacher.

To address this challenge, we introduce our distillation methodology, `SIKeD`: **S**elf-guided **I**terative **K**nowledge **D**istillation. The process begins with the LLM teaching the smaller model to approach tasks using a variety of reasoning strategies, providing a strong foundation for the smaller model to understand different problem-solving approaches. However, due to inherent biases and its limited capacity, the smaller model may still struggle to match the LLM's distribution of strategy choices effectively. To resolve this, we take inspiration from constructivist learning theory (Narayan et al., 2013), where the learner builds knowledge during the "assimilation phase" and refines their understanding during the "accommodation phase" to incorporate new insights. We propose generating outputs using the smaller model in an on-policy setup and selecting the best strategies for the task. By mixing the LLM-generated data with self-generated outputs, we leverage the strengths of both datasets. We iteratively fine-tune the smaller model allowing it to recognize strategies that it learned from the LLM but did not initially apply. With this approach, we align the smaller model with its own learned knowledge rather than forcing its distribution to mirror that of the LLM's.

Our proposed method extends beyond traditional one-step distillation, as each iteration of `SIKeD` leads to an updated policy that better grasps new information. We repeat multiple iterations of `SIKeD` based on the accuracy-cost tradeoff (does the improvement justify the cost of another iteration), allowing for continuous refinement and improvement of the model's reasoning capabilities. We demonstrate the effectiveness of `SIKeD` on several mathematical reasoning tasks using models with 7 billion parameters or less.[2] On four mathematical datasets—GSM8K (Cobbe et al., 2021), SVAMP (Patel et al., 2021), ASDiv (Miao et al., 2020), and MultiArith (Roy and Roth, 2015)—our approach achieves improvements of up to +5 points over traditional distillation strategies. Additionally, we show that multiple rounds of `SIKeD` allow the model to select the appropriate strategy for a given problem, while traditional distillation using LLM's data tends to leave it biased.

2 Preliminaries: LLM based Distillation

Problem Setup We consider the standard setup for LLM-based distillation (also referred to as *knowledge distillation*), where distillation data is sampled from a larger model with intermediate reasoning steps and a smaller distilled model is fine-tuned on the data (Shridhar et al., 2023; Magister et al., 2023). Two auto-regressive sequence models are involved in the process: a larger model or the LLM denoted as p_L and a smaller model to be distilled as p^{θ}_{sm} (with learnable parameters θ). In this work, we consider a reasoning distillation setup where the distillation dataset $\mathcal{D}$ consists of pairs of math questions q_i and their numerical answers a_i, $i \in \{1, \ldots, n\}$. Our work focuses on

[2]We acknowledge that "smaller model" is a relative term, and we consider models with 7 billion parameters or less to be smaller models.

improving reasoning in smaller models by teaching them to utilize a variety of reasoning strategies. We consider three reasoning strategies in this work: Chain-of-Thought (CoT), Least-to-Most (L2M), and Program-of-Thought (PoT). For each question q_i and a specific *reasoning strategy*, denoted as $s \in S$, we generate the reasoning chain (or *rationale*), denoted as r_i leading to the final answer: $r_i \sim p_L(. \mid \mathtt{pr}_s, q_i)$, where, $\mathtt{pr}_s$ represents the strategy-specific prompt. The prompts used for the generation of reasoning chains are provided in Appendix A.

2.1 LLM based Distillation

We begin by creating an initial training dataset $\mathcal{D}_{\mathsf{LLM}}$ consisting of a quadruple of $\{q_i, a_i, s, r_i\}$ for each data point. We perform a data filtering by extracting the final answer $\hat{a}_i$ from the generated rationale r_i and comparing it with the ground truth answer a_i. We discard all samples that do not match, i.e., we keep samples where $\hat{a}_i = a_i$. This filtering process eliminates incorrect rationales, ensuring that only high-quality data is used for distilling the smaller models.

We start the distillation process by training the smaller model with the created dataset $\mathcal{D}_{\mathsf{LLM}}$. The question q_i is provided as input, and the smaller model p^{θ}_{sm} (with learnable parameters θ) is first instructed to generate the strategy s, followed by the rationale r_i that leads to the final answer a_i. The loss $\mathcal{L}_{\mathsf{L}}(\theta)$ is defined as:

$$\mathcal{L}_{\mathsf{L}}(\theta) = -\,\mathbb{E}_{(q_i,s,r_i)\sim\mathcal{D}_{\mathsf{LLM}}}\Bigg[\log\, p^{\theta}_{\mathsf{sm}}\,(s \mid q_i, I) + \sum_{t=1}^{M} \log p^{\theta}_{\mathsf{sm}}\,(r_{i,t} \mid r_{i,<t}, s, q_i, I)\Bigg]$$

where M represents the number of tokens decoded over time t in an autoregressive manner, and I is the instruction used during fine-tuning. Note that this is analogous to traditional knowledge distillation from LLMs except that we make a strategy choice before generating rationales.

Limitations of this standard distillation setup Training solely on LLM-generated data $\mathcal{D}_{\mathsf{LLM}}$ can lead to a distribution mismatch between the training data and the smaller model's output distribution. Specifically, the larger model due to its larger capacity, may produce correct reasoning across multiple strategies that the smaller model can find difficult to replicate directly (Agarwal et al., 2024). A comparison of the strategy selected by the LLM and the smaller model on 1K samples is presented in Figure 1. The smaller model performs poorly when generating outputs on its own, as the training data distribution $P_{\mathsf{train}}(x)$ is different from the model's output distribution $P^{\theta}_{\mathsf{sm}}(x)$ as $P^{(1)}_{\mathsf{train}}(x) = P_{\mathrm{LLM}}(x)$, where x represents the samples (q_i, s, r_i), and $P_{\mathrm{LLM}}(x)$ is the distribution of the data generated by the LLM p_L.

Proposed Solution To mitigate the distributional shift in strategy choice between the LLM and the smaller model, we propose to incorporate the smaller model's correct outputs into the training data. This *self-guided* training with data mixing aligns the training data distribution more closely with the smaller model's output distribution, making learning more effective. A visualization of the data mixing approach is presented in Figure 2 that demonstrates that data mixing reduces the distribution shift, bringing the LLM and the smaller model's output distribution closer. This allows the smaller model to choose the right strategy for a given task, much like the LLM.

3 SIKeD: Self-guided Iterative Knowledge Distillation

We propose SIKeD, an *iterative* training procedure where smaller models can take advantage of their own generations to refine their strategy choices for a given task. In a nutshell, we generate data from the smaller model, filter out the correct samples based on whether the generated solutions are correct, and mix this data with the LLM-generated data to adjust its strategy preferences. The smaller distilled model is used to iteratively generate data in an on-policy setting where it updates itself by leveraging both the LLM data and its own generations. This iterative process allows the smaller model to improve its reasoning abilities and strategy selection over time by leveraging the LLM's knowledge and prior learning. The following paragraphs discuss the steps involved in our proposed iterative distillation methodology and the training objective.

Data generation For each question q_i and its associated reasoning strategy s, we first generate K rationales using the current smaller model p^{θ}_{sm} as: $r_i^{(k)} \sim p^{\theta}_{\mathsf{sm}}(\cdot \mid s, q_i, I), \quad$ for $k = 1, \ldots, K$. Note that we generate multiple samples K as the likelihood of a correct answer being present in one

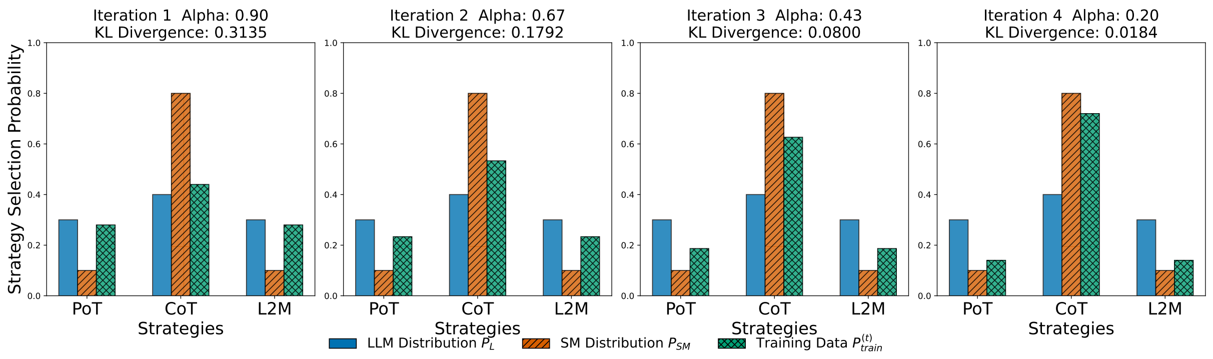

Figure 2: **Alignment of the smaller model's strategy distribution with the LLM over iterations**. Each subplot represents an iteration in the training process, showing the probability distributions over reasoning strategies: PoT, L2M, and CoT. The blue bars depict the LLM's distribution P_L, while the orange bars represent the smaller model's distribution P_{SM}, which is biased towards CoT. The green bars show the training data distribution $P_{\text{train}}^{(t)}$, a mixture of P_L and P_{SM} weighted by the mixing rate α. As α decreases over iterations (from 0.90 to 0.20), $P_{\text{train}}^{(t)}$ shifts from being similar to the LLM's distribution towards the smaller model's distribution. The KL divergence between the training data and the smaller model distributions decreases accordingly, indicating increased similarity.

of the rationales increases significantly with additional generations for smaller models (Jain et al., 2023; Wang et al., 2023).

Data Filtering Next, we extract the predicted answer $\hat{a}_i^{(k)}$ from each rationale $r_i^{(k)}$ and compare it with the ground truth a_i. We collect the correct samples, where $\hat{a}_i^{(k)} = a_i$, into a new dataset $\mathcal{D}_{\texttt{self}} = \left\{ \left(q_i, s, r_i^{(k)}\right) \middle| \hat{a}_i^{(k)} = a_i \right\}$.

Data mixing We combine the LLM-generated dataset $\mathcal{D}_{\texttt{LLM}}$ with the self-generated dataset $\mathcal{D}_{\texttt{self}}$ to form the mixed dataset $\mathcal{D}_{\texttt{mix}} = \mathcal{D}_{\texttt{LLM}} \cup \mathcal{D}_{\texttt{self}}$. Note that, we do not always use all the data from LLM in $\mathcal{D}_{\texttt{mix}}$, and study two variations: *All* when all LLM data is used in $\mathcal{D}_{\texttt{mix}}$, and *Adaptive* when only queries that have no correct generations in $\mathcal{D}_{\texttt{self}}$ are taken from $\mathcal{D}_{\texttt{LLM}}$. *Adaptive* uses less generated data from the LLM, resulting in more computationally efficient training.
The corresponding training data distribution changes to a mixture of the LLM data distribution and the smaller model's output distribution with correct samples:

$$P_{\text{train}}^{(2)}(x) = \alpha P_{\text{LLM}}(x) + (1-\alpha) P_{\text{sm}}^{\theta}(x),$$

where $\alpha = \frac{|\mathcal{D}_{\texttt{LLM}}|}{|\mathcal{D}_{\texttt{LLM}}| + |\mathcal{D}_{\texttt{self}}|}$ serves as a normalized mixing rate between the two datasets.

Training objective By including $\mathcal{D}_{\texttt{self}}$ in the training data, we reduce the divergence between $P_{\text{train}}^{(2)}(x)$ and the model's output distribution $P_{\text{sm}}^{\theta}(x)$, thus minimizing the distribution shift and improving training effectiveness of choosing the right strategy for a given task.
We continue training the smaller model on $\mathcal{D}_{\texttt{mix}}$ using the following loss function:

$$\mathcal{L}_{\texttt{mix}}(\theta) = -\,\mathbb{E}_{(q_i,s,r_i)\sim\mathcal{D}_{\texttt{mix}}} \Bigg[\log p_{\text{sm}}^{\theta}\,(s \mid q_i, I) + \sum_{t=1}^{M} \log p_{\text{sm}}^{\theta}\,(r_{i,t} \mid r_{i,<t}, s, q_i, I) \Bigg]$$

The expected loss over the training data is:

$$\mathcal{L}_{\texttt{mix}}(\theta) = -\mathbb{E}_{x\sim P_{\text{train}}^{(2)}(x)} \left[\log p_{\text{sm}}^{\theta}(x) \right]$$

where $x = (q_i, s, r_i)$, and $p_{\text{sm}}^{\theta}(x)$ denotes the probability assigned by the model to the sample x.

Analogous to minimizing the KL divergence Mixing the data is analogous to minimizing the Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951) between the training data distribution $P_{\text{train}}^{(2)}(x)$ and the model's output distribution $P_{\text{sm}}^{\theta}(x)$:

$$D_{\text{KL}}(P_{\text{train}}^{(2)}(x) \parallel P_{\text{sm}}^{\theta}(x)) = \sum_x P_{\text{train}}^{(2)}(x) \log \frac{P_{\text{train}}^{(2)}(x)}{P_{\text{sm}}^{\theta}(x)}$$

As we include more self-generated data, $(1-\alpha)$ increases, and $P_{\text{train}}^{(2)}(x)$ becomes closer to $P_{\text{sm}}^{\theta}(x)$. This reduces the KL divergence and aligns the training data distribution with the model output distribution, leading to more effective learning. Figure 2 demonstrates that as the value of α decreases, the distribution of the training data strategy choices aligns with the smaller model with correct solutions, with a reduction in their KL value over iterations. This allows the smaller model to better capture the strategy distribution of the larger model.

3.1 Iterative Self-Training of SIKeD

We repeat the data generation, filtering, mixing, and training steps iteratively. In each iteration t, the smaller model potentially generates new correct rationales that are added to the training data. The training data distribution at iteration t becomes:

$$P^{(t)}_{\text{train}}(x) = \alpha^{(t)} P_{\text{LLM}}(x) + (1-\alpha^{(t)}) P^{\theta^{(t-1)}}_{\text{sm}}(x),$$

where $\theta^{(t-1)}$ are the model parameters from the previous iteration, and $\alpha^{(t)}$ is updated based on the sizes of $\mathcal{D}_{\texttt{LLM}}$ and $\mathcal{D}^{(t)}_{\texttt{self}}$ at iteration t. Note that the generated samples from the smaller model automatically govern the value of $\alpha^{(t)}$ based on the size of $\mathcal{D}^{(t)}_{\texttt{self}}$.

This iterative process continues until the model's performance converges or a predefined stopping criterion is met. Over multiple iterations, the model's output distribution $P^{\theta^{(t)}}_{\text{sm}}(x)$ gradually improves, and the training data distribution becomes increasingly aligned with it. We present an end-to-end training methodology in Algorithm 1.

Algorithm 1 **SIKeD: Self-guided Iterative Knowledge Distillation**

```
Input: $\mathcal{D}$: Reasoning dataset with questions $\{q_i\}_{i=1}^N$ and answers $\{a_i\}_{i=1}^N$, $\mathcal{D}_{\texttt{LLM}}$: Reasoning dataset generated using the LLM with questions $\{q_i\}$, answers $\{a_i\}$, strategy $\{s\}$, rationales $\{r_i\}$, $S$: Set of reasoning strategies, $I$ : instruction, $p_L$: LLM for rationale generation , $p_{\texttt{sm}}^{\theta^{(0)}}$: Smaller model with initial parameters $\theta^{(0)}$ , $K$: Number of samples per question and strategy, $T$: Maximum number of iterations, Variation: All or Adaptive,
// LLM-Based Distillation
Train $p_{\texttt{sm}}^{\theta^{(0)}}$ on $\mathcal{D}_{\texttt{LLM}}$ by minimizing $\mathcal{L}_{\texttt{L}}(\theta^{(0)})$ ( (2.1))
// SIKeD: Self-guided Iterative Knowledge Distillation
for iteration $t = 1$ to $T$ do
    Initialize dataset $\mathcal{D}^{(t)}_{\texttt{self}} \leftarrow \emptyset$
    for each question $q_i \in \mathcal{D}$ do
        for each strategy $s \in S$ do
            for $k = 1$ to $K$ do
                Generate rationale $r_i^{(k)}$ using $p_{\texttt{sm}}^{\theta^{(t-1)}}$: $r_i^{(k)} \sim p_{\texttt{sm}}^{\theta^{(t-1)}}(\cdot \mid s, q_i, I)$
                Extract answer $\hat{a}_i^{(k)}$ from $r_i^{(k)}$
                if $\hat{a}_i^{(k)} = a_i$ then
                    Add $(q_i, s, r_i^{(k)})$ to $\mathcal{D}^{(t)}_{\texttt{self}}$
                end
            end
        end
    end
    if Variation is All then
        Combine datasets: $\mathcal{D}^{(t)}_{\texttt{mix}} = \mathcal{D}_{\texttt{LLM}} \cup \mathcal{D}^{(t)}_{\texttt{self}}$
    else
        Identify questions with no correct self-generated rationales:
          $\mathcal{I} = \{i \mid \text{no correct } r_i^{(k)} \text{ in } \mathcal{D}^{(t)}_{\texttt{self}}\}$
          Include corresponding LLM data: $\mathcal{D}^{(t)}_{\texttt{LLM}} = \{(q_i, s, r_i) \in \mathcal{D}_{\texttt{LLM}} \mid i \in \mathcal{I}\}$
          Combine datasets: $\mathcal{D}^{(t)}_{\texttt{mix}} = \mathcal{D}^{(t)}_{\texttt{LLM}} \cup \mathcal{D}^{(t)}_{\texttt{self}}$
    end
    Update $\alpha^{(t)} = \frac{|\mathcal{D}^{(t)}_{\texttt{LLM}}|}{|\mathcal{D}^{(t)}_{\texttt{LLM}}| + |\mathcal{D}^{(t)}_{\texttt{self}}|}$
    Retrain $p_{\texttt{sm}}^{\theta^{(t)}}$ on $\mathcal{D}^{(t)}_{\texttt{mix}}$ by minimizing $\mathcal{L}^{(t)}_{\texttt{mix}}(\theta^{(t)})$ ( (3))
end
Output: Updated smaller model $p_{\texttt{sm}}^{\theta^{(T)}}$
```

4 Experimental Details

Dataset Our work demonstrates the effectiveness of selecting an appropriate strategy for a given task. We consider multi-step mathematical reasoning datasets in our work, as various strategies can solve the task fairly well. We trained SIKeD on the GSM8K training set (Cobbe et al., 2021), which includes 7,473 samples, and tested it on the corresponding test set of 1,319 samples. To assess the domain transferability of our distillation method, we also evaluated it on three additional mathematical datasets: SVAMP (Patel et al., 2021) with 1,000 samples, ASDiv (Miao et al., 2020) with 2,300 test samples, and MultiArith (Roy and Roth, 2015) with 180 samples. As the GSM8K training set was used to train the smaller model, we classify it as an *in-distribution* dataset. In contrast, no training data from SVAMP, ASDiv, or MultiArith was used, as they were exclusively employed for testing purposes and thus considered *out-of-distribution*.

Implementation Details We used the Llama3 70B model (Dubey et al., 2024) as the large language model (LLM) to generate the rationales. We performed distillation on different smaller models ranging from 0.5B to 7B parameters, including Qwen2 0.5B (Bai et al., 2023), Qwen2 1.5B (Bai et al., 2023), SmolLM 1.7B (Hugging Face, 2023), Gemma 2B (Team et al., 2024), and Gemma 7B (Team et al., 2024). All smaller models were fine-tuned using LoRA (Hu et al., 2022) with a rank of 16, and alpha of 32. We used a learning rate of 3e-4 for Qwen models with a cyclic scheduler, while we set 2e-4 as the learning rate for other models and used a linear scheduler. We train all models for 3 epochs. We implemented all our experiments using the Unsloth FastLanguageModel (Unslothai, 2023) and used the VLLM library (Kwon et al., 2023) for inference. We set the temperature $t = 0$ for data generation from the LLM while $t = 0.7$ was used for generating samples from the smaller model at

Dataset	Method	Gemma 7B	Gemma 2B	SmolLM 1.7B	Qwen 1.5B	Qwen 0.5B
GSM8K	CoT	67.40	36.54	16.38	55.57	36.47
	L2M	69.29	36.92	18.73	54.59	33.59
	PoT	71.34	44.05	23.73	64.22	41.62
	Combined	70.74	44.05	24.56	64.44	42.38
	SIKeD(Adaptive)	**73.84** (↑ +2.5)	**47.23** (↑ +3.2)	**27.98** (↑ +3.4)	**64.97** (↑ +0.3)	**43.14** (↑ +0.8)
	SIKeD(All)	71.42 (↑ +0.1)	45.26 (↑ +1.2)	27.75 (↑ +3.2)	64.14 (↓ -0.3)	43.06 (↑ +0.7)
ASDiv	CoT	68.76	54.01	30.37	68.76	54.66
	L2M	64.69	43.47	22.13	63.69	49.76
	PoT	67.85	58.13	43.77	66.94	56.83
	Combined	69.11	57.96	46.77	67.64	57.79
	SIKeD(Adaptive)	70.59 (↑ +1.5)	**59.05** (↑ +0.9)	47.20 (↑ +0.4)	**68.98** (↑ +1.3)	58.44 (↑ +0.7)
	SIKeD(All)	**70.76** (↑ +1.6)	58.00 (↓ -0.1)	**48.16** (↑ +1.4)	68.55 (↑ +0.9)	**58.61** (↑ +0.8)
MultiArith	CoT	98.33	87.22	58.89	**99.44**	83.89
	L2M	96.11	81.67	53.89	96.67	76.67
	PoT	98.89	90.56	61.11	95.56	92.22
	Combined	99.44	84.44	67.22	98.89	90.56
	SIKeD(Adaptive)	99.44 (-)	**91.11** (↑ +0.6)	**72.22** (↑ +5.0)	**99.44** (-)	**93.33** (↑ +1.1)
	SIKeD(All)	**100.0** (↑ +0.6)	88.89 (↓ -1.6)	**72.22** (↑ +5.0)	98.89 (↓ -0.5)	92.22 (-)
SVAMP	CoT	66.80	41.90	22.60	66.30	43.00
	L2M	64.80	31.60	17.90	62.30	44.60
	PoT	75.00	56.80	34.50	74.30	51.40
	Combined	69.40	56.20	35.90	73.20	51.40
	SIKeD(Adaptive)	72.90 (↓ -2.1)	**58.60** (↑ +1.8)	37.80 (↑ +1.9)	**75.40** (↑ +1.1)	51.70 (↑ +0.3)
	SIKeD(All)	**76.40** (↑ +1.4)	56.70 (↓ -0.1)	**39.50** (↑ +3.6)	73.50 (↓ -0.8)	**52.10** (↑ +0.7)

Table 1: Top-1 (maj@1) accuracy across four math datasets (rows) and five smaller models (columns). We report the performance for four baseline methods: CoT, L2M, PoT, Combined and compare them with two SIKeDvariants (Adaptive / All). Best performance in each row is in **bold**.

each iteration. We set the number of generated samples or K to 10. We report Top-1 accuracy (maj@1).

Often, distillation is performed using only a single strategy such as Chain-of-Thought (CoT) (Wei et al., 2022), Program-of-Thought (PoT) (Chen et al., 2023), Least-to-Most (L2M) (Zhou et al., 2023) or by combining different strategies (Chenglin et al., 2023; Zhu et al., 2024). For this reason, we treat single-strategy distillation (CoT, PoT, L2M) and "Combined" as baseline models in our work as shown in Table 1.

5 Results and Discussion

LLM Based Distillation We start by distilling smaller models using the reasoning dataset generated using the LLM in two variations: using data from a single strategy (CoT, PoT, or L2M), and a combination of all three strategies (referred to as "Combined"). Table 1 compares the accuracies of the approaches across four mathematical datasets. The "Combined" approach benefited smaller models, yielding slight improvements for the Qwen 0.5B, Qwen 1.5B, and SmolLM 1.7B models. However, it showed little to no improvement, and sometimes even worse performance, for the larger Gemma 2B and 7B models. This indicates that simply merging the distillation data for each strategy is not sufficient for effective multi-strategy distillation.

Consistent improvement across in-distribution dataset Compared to the traditional LLM-based distillation approaches, we observe consistent improvements with SIKeD across all models, ranging from 0.5B to 7B parameters as **shown in bold** in Table 1. On the in-distribution GSM8K dataset, both Gemma 2B and 7B show significant gains of +3.2 points and +2.5 points respectively (44.05 $\rightarrow$ 47.23 and 71.34 $\rightarrow$ 73.84, respectively). Similarly, SmolLM showed the largest improvement of +3.4 points (24.56 $\rightarrow$ 27.98). Similarly, the smaller Qwen models see gains of +0.5 points for the larger variant (1.5B) and +0.8 points for the smaller variant (0.5B).

SIKeD performs well on out-of-distribution datasets For the out-of-distribution datasets, there is a steady improvement on the ASDiv dataset, with Gemma 7B gaining +1.6 points (69.11 $\rightarrow$ 70.76), +0.9 points for Gemma 2B (58.13 $\rightarrow$ 59.05), +1.4 points for SmolLM (46.77 $\rightarrow$ 48.16), +1.3 points for Qwen 1.5B (67.64 $\rightarrow$ 68.98), and +0.8 points for Qwen 0.5B (57.79 $\rightarrow$ 58.61). A similar trend is seen for the MultiArith dataset, where SmolLM shows the largest gain of +5 points. It is followed by Qwen 0.5B with +1.1 points, while

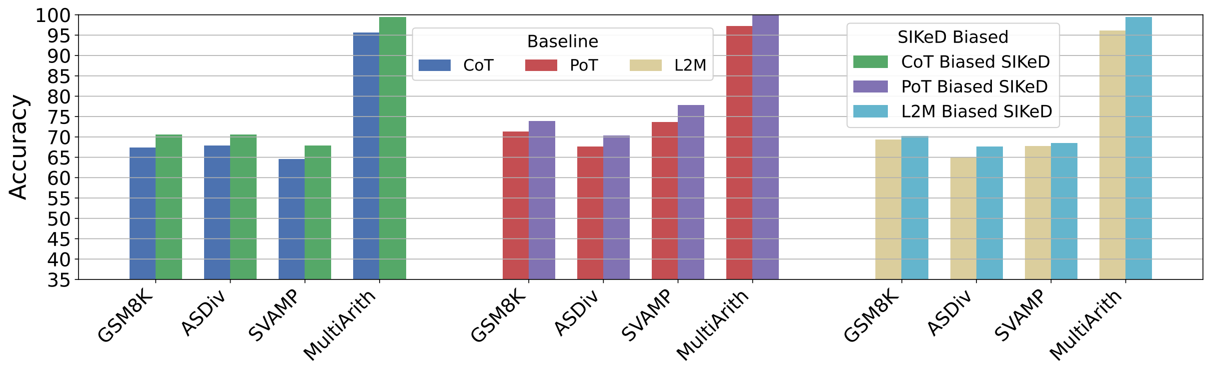

Figure 3: Accuracy comparison between single distillation strategies of CoT, PoT, and L2M with SIKeD biased training using the same strategy using the Gemma 7B model.

other models outperform the baseline. In particular, Gemma 7B achieves a perfect score of 100. The results are similar for the SVAMP dataset, with Qwen 0.5B, Qwen 1.5B, SmolLM 1.7B, Gemma 2B, and Gemma 7B gaining +0.7, +1.1, +3.6, +1.8 and +1.4 points, respectively.

Biasing `SIKeD` in favor of our strategy of choice For some tasks, one strategy might be better than the others due to its performance, lower cost, or better suitability for some use cases (for example, PoT is significantly better for SVAMP compared to other strategies). In such cases, it would be beneficial to bias the selection towards that strategy[3]. This can be done by taking only the sample from our choice of biased strategy when more than one strategy is correct from the model-generated samples. For example, if for a given data point, a smaller model samples both CoT and PoT correctly, and our biased strategy choice is PoT, we will ignore the CoT output and take only the PoT. Figure 3 compares the individual distillation strategy with the biased SIKeD. Using Gemma 7B as a smaller model across all datasets, SIKeD outperforms individual distillation strategies by a margin of 2-4 points, highlighting the effectiveness of SIKeD over other distillation approaches.

How many iterations to run for `SIKeD` With each iteration of SIKeD, the model learns to solve a task using different strategies and adjusts its strategy choice for a given task. This allows for continuous training of SIKeD. Figure 4 illustrates the accuracy improvements across iterations for the Gemma 2B model on various datasets. The iterative training is stopped when accuracy shows only marginal improvements or declines. In our experiments, three iterations have consistently proven to be the optimal balance across different models and datasets.

[3]Note that this differs from the already biased selection of the smaller model, as our biased strategy may not be the default biased choice of the smaller model.

How the strategy distribution changes over iterations Figure 5 illustrates the strategy distribution across different iterations for the GSM8K dataset using the SmolLM 1.7B model. Iteration 0 represents the baseline "combined" training from Table 1, and as expected, the smaller model is initially biased towards one strategy (PoT in this case). Iterations 1, 2, and 3 show the model's progression using SIKeD, where it learns to diversify and select the suitable strategy for the given problem. Notably, while PoT remains the dominant strategy, the model improves its usage of the other two strategies—CoT and L2M—which were absent in the biased baseline. This diversification of strategies results in an overall gain of +3 points over the baseline.

Training from the last checkpoint vs training from pre-trained checkpoint In our work, we iteratively train from the last checkpoint using on-policy training as we expect continuous improvements in the model performance with a newly learned strategy. However, an alternative approach uses off-policy training (training the pre-trained model at each iteration) to achieve strong performance (Gulcehre et al., 2023). We compared on-policy training (our proposed approach) with off-policy training (as in Gulcehre et al. (2023)), utilizing both LLM-generated and self-generated data, and observed a notable decrease in the overall accuracy with off-policy training. Note that we used all of the LLM data at each iteration for off-policy

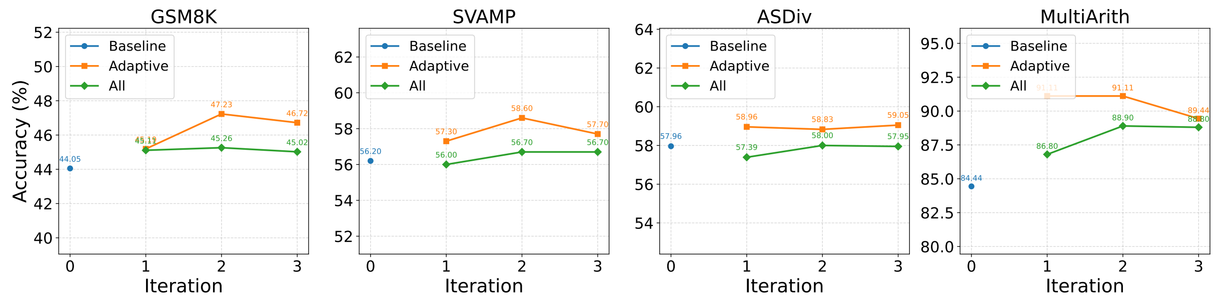

Figure 4: Iterative accuracy comparison for the Gemma 2B model across all datasets. The process is stopped when the gains diminish or when it is no longer cost-effective to continue. *Baseline* represents the "Combined" results from Table 1.

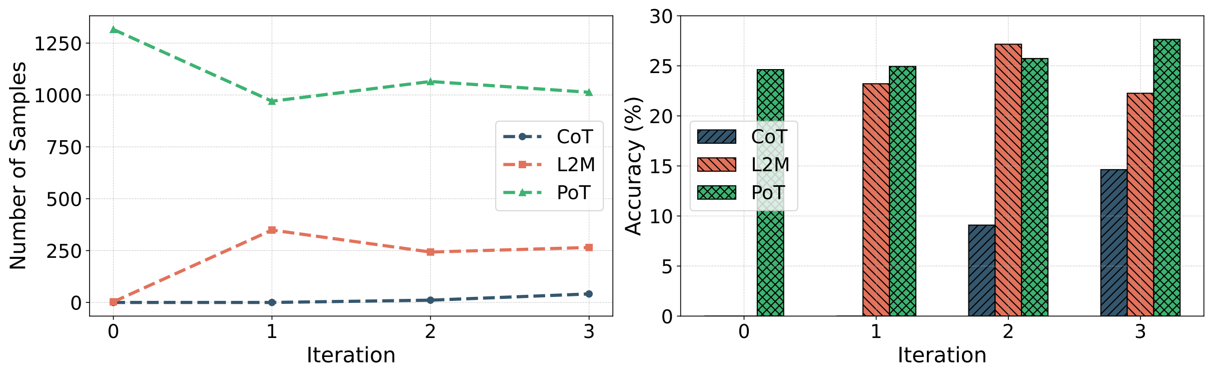

Figure 5: Strategy distribution over iterations for GSM8K dataset using SmolLM 1.7B model.

training as the training is done on the pre-trained model. On the GSM8K dataset, our on-policy approach outperformed off-policy training by +6 points (45.26 vs 38.90) using the Gemma 2B model. A similar trend was observed on out-of-distribution datasets, where SIKeD outperformed with a gain of +4-7 points on both the SVAMP and ASDiv datasets (58.6 vs 51.3 for SVAMP and 59.05 vs 55.44 for ASDiv) and a gain of +2 points on the MultiArith dataset (91.11 vs 88.33).

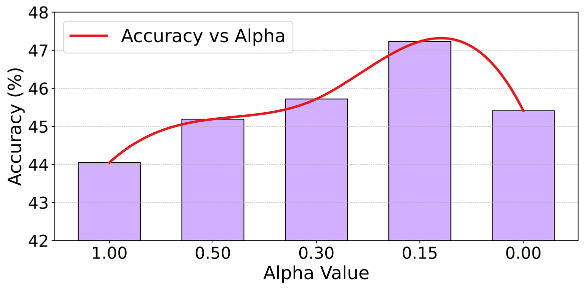

Figure 6: Accuracy comparison with different data mix controlled by α. $\alpha = 1$ means all data is from the LLM only, while $\alpha = 0$ means only smaller model data is used.

Is data mixing better than LLM-based distillation or self-distillation Mixing data from the smaller model with the LLM helps in bridging the distributional gap between the LLM and the smaller model. We explore the role of the data mixing rate alpha (α) to validate our hypothesis regarding data mixing. When α is set to 1, only LLM-generated data is used, while at the other extreme, $\alpha = 0$ means only data generated by the smaller model is used. As shown in Figure 6, an α value between 0 and 1 performs better than either of the extreme cases.

Qualitative analysis demonstrating that smaller models learn to choose the right strategy over iterations The increase in the performance of smaller models can be attributed to the change in the distribution of reasoning strategies as the model becomes more aligned with its strategy choices. Figure 10 shows that a model can solve a given problem in the third iteration by switching the reasoning strategy that it initially struggled with. Furthermore, Figure 11 shows an example of a model correcting its reasoning chain over time due to its improved reasoning capabilities.

6 Related Work

Knowledge Distillation for Reasoning Tasks Knowledge distillation (Buciluǎ et al., 2006; Hinton et al., 2015) is a widely-used technique for transferring knowledge from a large language model (LLM) to a smaller model. Previous research has focused on transferring intermediate reasoning steps to smaller models, either step-by-step (Shridhar et al., 2023; Magister et al., 2023; Hsieh et al., 2023) or by combining multiple strategies simultaneously (Chenglin et al., 2023; Zhu et al., 2024). These approaches can be viewed as aggregating diverse data sources for distillation, similar to the LLM data approach in our work. In contrast, (Hahn and Choi, 2019) and (Xu et al., 2020) focus on *self-distillation*, where a model improves its performance without external data or knowledge. Specifically, (Hahn and Choi, 2019) leverages word embeddings, while (Xu et al., 2020) uses temporal model checkpoints as a proxy for ground truth. However, both approaches rely solely on data generated by the smaller model and exclude LLM data. On the other extreme is iteratively updating the teacher's data based on student mistakes to distill reasoning in smaller models iteratively (Jain et al., 2025). Our method balances these two extremes by using LLM data to learn multiple strategies and self-generated data to optimize for the right strategy choice.

Self-learning Previous studies, such as (He et al., 2020; Sun et al., 2021; Gulcehre et al., 2023; Liu et al., 2024), have shown the effectiveness of the *self-training paradigm* in NLP tasks but are limited by the choice of tasks. While ReST (Gulcehre et al., 2023) uses off-policy training, we find on-policy training more suitable for our case regarding data efficiency and performance. On-policy training also allows a better choice of learning strategies, since the model can use its most recent learning. Agarwal et al. (2024) introduces Generalized Knowledge Distillation (GKD), an on-policy training method that aligns the distributions of large language models (LLMs) and smaller models by incorporating output sequences sampled from the student during training. However, the task was limited to the distribution alignment and not to aligning the strategy choices in a multi-strategy distillation. Simply applying GKD would not address this issue, as it would force the smaller model to learn all strategies, which is impractical given its limited capacity.

Finally, we compare our distillation strategies with LLM-based distillation using both individual strategies (Shridhar et al., 2023; Magister et al., 2023; Hsieh et al., 2023) and a combination of several strategies at once (Chenglin et al., 2023; Zhu et al., 2024).

7 Conclusion

We propose SIKeD: **S**elf-guided **I**terative **K**nowledge **D**istillation, which addresses the challenge of distilling multistep reasoning skills from large language models (LLMs) to smaller models. Unlike traditional methods, which often leave smaller models biased towards a single strategy, SIKeD uses iterative *self-guided* training, combining LLM and self-generated data to improve overall reasoning in smaller models. We evaluate our approach across various mathematical reasoning datasets and demonstrate that SIKeD improves the ability of smaller models to handle complex reasoning, achieving significant performance gains.

Limitations

Although SIKeD shows consistent performance gains across five models and four mathematical datasets, we note that SIKeD is dependent on the limitations of reasoning strategies. For instance, mathematical questions whose reasoning chains could not be framed in PoT, will most likely give less substantial increments using SIKeD as it has fewer strategies to utilize.

Ethical Considerations

Ethical considerations should be taken into account when deploying language models in real-world applications. We do not foresee any additional concerns emanating from this work or any potential risks.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint*.

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. 2024. On-policy distillation of language models: Learning from self-generated mistakes.

In *The Twelfth International Conference on Learning Representations*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint*.

Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, Philadelphia PA USA. ACM.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*.

Li Chenglin, Chen Qianglong, Wang Caiyu, and Zhang Yin. 2023. Mixed distillation helps smaller language model better reasoning. *arXiv preprint*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint*.

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*.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. 2023. Reinforced self-training (rest) for language modeling. *arXiv preprint*.

Isabel Gómez-Veiga, José O. Vila Chaves, Gonzalo Duque, and Juan A. García Madruga. 2018. A new look to a classic issue: Reasoning and academic achievement at secondary school. *Frontiers in Psychology*, 9.

Sangchul Hahn and Heeyoul Choi. 2019. Self-knowledge distillation in natural language processing. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 423–430.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2020. Revisiting self-training for neural sequence generation. In *International Conference on Learning Representations*.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex 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. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017. Association for Computational Linguistics.

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*.

Hugging Face. 2023. smol-llm: Train a small llm from scratch. `https://huggingface.co/blog/smollm`. Accessed: 2024-09-23.

Kushal Jain, Piyushi Goyal, and Kumar Shridhar. 2025. Undo: Understanding distillation as optimization. *arXiv preprint*.

Kushal Jain, Moritz Miller, Niket Tandon, and Kumar Shridhar. 2023. First-step advantage: Importance of starting right in multi-step math reasoning. *ArXiv*, abs/2311.07945.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

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*.

Rongxing Liu, Kumar Shridhar, Manish Prajapat, Patrick Xia, and Mrinmaya Sachan. 2024. Smart: Self-learning meta-strategy agent for reasoning tasks. *arXiv preprint*.

Qing Lyu, Kumar Shridhar, Chaitanya Malaviya, Li Zhang, Yanai Elazar, Niket Tandon, Marianna Apidianaki, Mrinmaya Sachan, and Chris Callison-Burch. 2024. Calibrating large language models with sample consistency. *arXiv preprint*.

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. Association for Computational Linguistics.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing English math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984. Association for Computational Linguistics.

Ratna Narayan, Cynthia Rodriguez, Juan Araujo, Ali Shaqlaih, and Glenda Moss. 2013. Constructivism—constructivist learning theory. *IAP Information Age Publishing*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094. Association for Computational Linguistics.

Dean A Pomerleau. 1991. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97.

Stephane Ross and Drew Bagnell. 2010. Efficient reductions for imitation learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 661–668. PMLR.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752. Association for Computational Linguistics.

Kumar Shridhar, Jakub Macina, Mennatallah El-Assady, Tanmay Sinha, Manu Kapur, and Mrinmaya Sachan. 2022. Automatic generation of socratic subquestions for teaching math word problems. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4136–4149. Association for Computational Linguistics.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2023. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7059–7073, Toronto, Canada. Association for Computational Linguistics.

Haipeng Sun, Rui Wang, Kehai Chen, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2021. Self-training for unsupervised neural machine translation in unbalanced training data scenarios. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3975–3981. Association for Computational Linguistics.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Unslothai. 2023. Unsloth. `https://github.com/unslothai/unsloth`. GitHub repository.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Yige Xu, Xipeng Qiu, Ligao Zhou, and Xuanjing Huang. 2020. Improving BERT Fine-Tuning via Self-Ensemble and Self-Distillation. *arXiv preprint*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2024. Distilling mathematical reasoning capabilities into small language models. *arXiv preprint arXiv:2401.11864*.

A Appendix

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

Instruction: Solve the given math problem step by step. Put your final answer after 'Final answer:'.

Input: John buys 3 dress shirts. They sell for $20 each. He also has to pay 10% tax on everything. How much did he pay in total?

Response: The shirts cost 3*$20=$<<3*20=60>>60 before tax The tax cost $60*.1=$<<60*.1=6>>6 So in total they paid $60+$6=$<<60+6=6>> 66 Final Answer: 66<eos>

[7 more examples randomly sampled from the training set]

Input: Thomas is training at the gym to prepare for a competition. He trained for 5 hours every day for a month (30 days). If he continues to train for the next 12 days, how many hours will he spend on training in total?

Response: Model generated response ..

Figure 7: 8-shot Chain of Thought demonstration.

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

Instruction: Solve the given math problem by decomposing it into smaller, manageable sub-questions. Put your final answer after 'Final answer: '.

Input: John buys 3 dress shirts. They sell for $20 each. He also has to pay 10% tax on everything. How much did he pay in total?

Response: Sub-question 1: How much did John spend on shirts?
Answer to Sub-question 1: 3*$20=$<<3*20=60>>60
Sub-question 2: How much did John spend on tax?
Answer to Sub-question 2: $60*.1=$<<60*.1=6>>6
Sub-question 3: How much did John spend in total?
Answer to Sub-question 3: $60+$6=$<<60+6=66>>66
Final Answer: 66<eos>

[7 more examples randomly sampled from the training set]

Input: Thomas is training at the gym to prepare for a competition. He trained for 5 hours every day for a month (30 days). If he continues to train for the next 12 days, how many hours will he spend on training in total?

Response: Model generated response ..

Figure 8: 8-shot Least to Most demonstration.

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

Instruction: Solve the given math problem by writing a python program. Store your result as a variable named 'answer'.

Input: John buys 3 dress shirts. They sell for $20 each. He also has to pay 10% tax on everything. How much did he pay in total?

```
Response: total_shirts = 3
cost_of_one_shirt = 20
total_cost_shirts = total_shirts * cost_of_one_shirt
tax_rate = 0.1
tax_amount = tax_rate * total_cost_shirts
total_cost = total_cost_shirts + tax_amount
answer = total_cost <eos>
```

[7 more examples randomly sampled from the training set]

Input: Thomas is training at the gym to prepare for a competition. He trained for 5 hours every day for a month (30 days). If he continues to train for the next 12 days, how many hours will he spend on training in total?

Response: Model generated response ..

Figure 9: 8-shot Program of Thought demonstration.

Question There are twice as many boys as girls at Dr. Wertz's school. If there are 60 girls and 5 students to every teacher, how many teachers are there?

Baseline
girls = 60
boys = 2 $*$ girls
teachers = girls$/$5
answer = teachers
Final Answer: 12 ✗

Iteration1
girls = 60
boys = 2 $*$ girls
teachers = girls$/$5
answer = teachers
Final Answer: 12 ✗

Iteration2
girls = 60
boys = 2 $*$ girls
total_students = girls $+$ boys
teachers_per_student = 5
total_teachers = total_students * teachers_per_student
answer = total_teachers
Final Answer: 900 ✗

Iteration3
Sub-question 1: How many boys are there?
Answer to Sub-question 1: $2 * 60 =<< 2 * 60 = 120 >> 120$
Sub-question 2: How many students are there in total?
Answer to Sub-question 2: $60 + 120 =<< 60 + 120 = 180 >> 180$
Sub-question 3: How many teachers are there?
Answer to Sub-question 3: $180/5 =<< 180/5 = 36 >> 36$
Final Answer: 36 ✓

Figure 10: A qualitative example showing how Gemma 2B shifts its strategy selection over iterations.

Question In a dance class of 20 students, 20% enrolled in contemporary dance, 25% of the remaining enrolled in jazz dance, and the rest enrolled in hip-hop dance. What percentage of the entire students enrolled in hip-hop dance?

Baseline
total_students = 20
percentage_enrolled_contemporary = 0.2
students_enrolled_contemporary = total_students $*$ percentage_enrolled_contemporary
remaining_students = total_students $-$ students_enrolled_contemporary
percentage_enrolled_jazz = 0.25
students_enrolled_jazz = remaining_students $*$ percentage_enrolled_jazz
remaining_students = remaining_students - students_enrolled_jazz
percentage_enrolled_hip_hop = 1 $-$ percentage_enrolled_contemporary $-$ percentage_enrolled_jazz
answer = percentage_enrolled_hip_hop
Final Answer: 0.55 ✗

Iteration1
total_students = 20
contemporary_students = total_students $*$ 0.2
remaining_students = total_students $-$ contemporary_students
jazz_students = remaining_students $*$ 0.25
hip_hop_students = remaining_students $-$ jazz_students
percentage_hip_hop = hip_hop_students$/$total_students $*$ 100
answer = percentage_hip_hop
Final Answer: 60 ✓

Figure 11: Over iteration, Gemma 2B model learned to fix its mistake using the same strategy