Babysitting a Small Language Model through One-Step Tree-of-Thoughts Knowledge Distillation

Anurag Renduchintala[1], **Adi Mahesh**[1], **Zichen Zhang**[1], **Zimo Si**[1], **Shangjun Meng**[1], **Samuel Fang**[1]
[1]University of Michigan
{ranurag, mahesha, zhangzzc, zimosi, shangjun, swfang}@umich.edu

Abstract

The growing computational and environmental costs of Large Language Models (LLMs) have driven the demand for Small Language Models (SLMs) that offer comparable reasoning capabilities. However, existing prompting methods, such as Chain-of-Thought (CoT) or Multi-Step Tree-of-Thoughts (ToT), often fail to work effectively with SLMs due to their limited context windows and capacity. In this paper, we propose a novel approach that simplifies the ToT framework into a One-Step ToT prompting method and leverages knowledge distillation to transfer ToT reasoning capabilities from LLMs to SLMs. Our methodology involves synthesizing a dataset by prompting a GPT-4o model with the One-Step ToT framework and fine-tuning an SLM, SmolLM-360M, using this dataset. By replacing the traditional multi-step prompts with a single, structured prompt, we enable the LLM to generate ToT-style reasoning in a more efficient format. The SLM is then fine-tuned on these outputs to emulate the reasoning process. Using the Game of 24 dataset as a benchmark, we demonstrate that this approach enables SLMs to achieve competitive reasoning performance over LLMs while maintaining computational efficiency.

Introduction

Large Language Models (LLMs) have revolutionized natural language processing (NLP), achieving remarkable success in tasks such as translation, summarization, and question-answering by capturing intricate patterns in human language (Zhao et al. 2023). These models are adept at generating coherent and contextually appropriate text through sequential token prediction, where each word is determined based on the preceding context. However, this left-to-right generation mechanism, while powerful, poses challenges for tasks that demand multi-step reasoning or complex decision-making.

The Multi-Step Tree-of-Thoughts (ToT) was proposed to improve logical multi-step reasoning in LLMs beyond simple prompting and other existing prompting frameworks by attempting to break through the limitations of LLMs' inherent linearity in output generation. Yao et al. 2023 successfully demonstrated that generating and exploring different thought branches allowed LLMs to achieve significant performance gains in tasks that require backtracking or exploration.

Despite these advancements, the immense computational and environmental costs associated with training and deploying LLMs underscore the need for more efficient alternatives (Bender et al. 2021). Small Language Models (SLMs) have emerged as a promising solution, offering a balance between performance and efficiency (Wang et al. 2024). However, SLMs struggle with complex reasoning tasks due to their reduced capacity, and unlike LLMs, advanced prompting techniques like CoT and Multi-Step ToT, which rely on iterative prompts and large context windows, are often ineffective for SLMs with limited resources.

In this paper, we first replicate the Multi-Step ToT (Yao et al. 2023) and then introduce a novel **One-Step ToT** to address these challenges. We successfully replicate Multi-Step ToT's performance advantage over CoT and traditional Input-Output (IO) prompting in solving the Game of 24, a challenging arithmetic reasoning task, and demonstrate improvements using GPT-4o (OpenAI 2024b) over the original GPT-4 used in prior work. Our One-Step ToT simplifies this Multi-Step ToT into a single prompt, making it suitable for SLM fine-tuning. Using this One-Step ToT, we distill the ToT-style reasoning capabilities of LLMs into SLMs through knowledge distillation. Using GPT-4o, we synthesize a dataset containing correct ToT-style reasoning and fine-tune a much smaller model, SmolLM-360M (Allal et al. 2024).

We evaluate our new One-Step ToT and fine-tuned SLM on the Game of 24. Our results show that when used on LLMs like GPT-4o, the One-Step ToT enables LLMs to perform better than those prompted with naive CoT. We further show that fine-tuned with a synthesized dataset, SLMs with only 360M parameters can achieve better performance on arithmetic reasoning than LLMs, effectively bridging the gap between efficiency and reasoning capability. The main contributions of this paper are as follows:

1. We replicated CoT and ToT performances using the latest GPT-4o on the Game of 24, achieving higher success rates than the GPT-4 in Yao et al. 2023.
2. We proposed One-Step ToT, a simplified prompting framework that integrates ToT reasoning into a single structured prompt, and showed its effectiveness over naive CoT. We demonstrated that after distilling ToT-style knowledge into an SLM like SmolLM-360M, the SLM can achieve significant improvements on the Game of 24 and rival LLMs like GPT-4o.

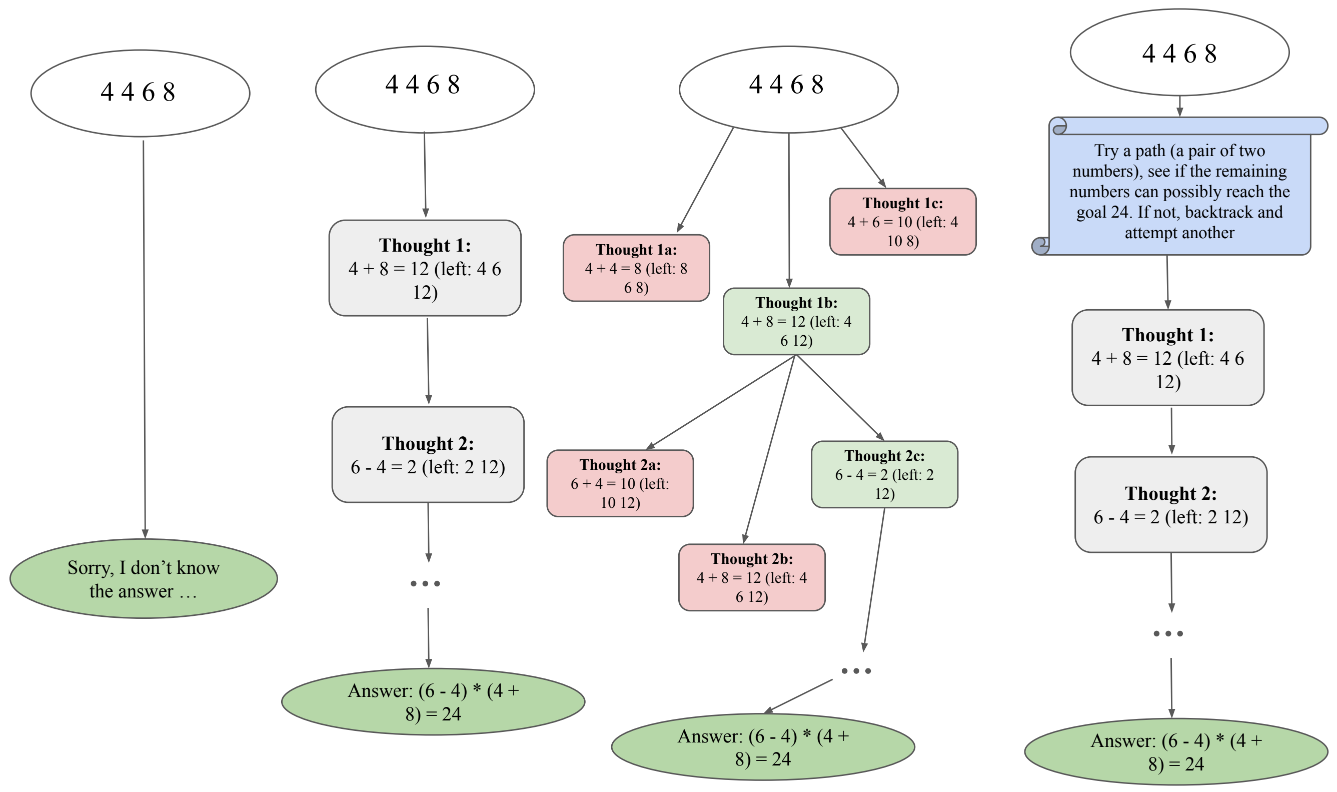
4 4 6 8
Sorry, I don't know the answer …
4 4 6 8
Thought 1:
4 + 8 = 12 (left: 4 6 12)
Thought 2:
6 - 4 = 2 (left: 2 12)
…
Answer: (6 - 4) * (4 + 8) = 24
4 4 6 8
Thought 1a:
4 + 4 = 8 (left: 8 6 8)
Thought 1b:
4 + 8 = 12 (left: 4 6 12)
Thought 1c:
4 + 6 = 10 (left: 4 10 8)
Thought 2a:
6 + 4 = 10 (left: 10 12)
Thought 2b:
4 + 8 = 12 (left: 4 6 12)
Thought 2c:
6 - 4 = 2 (left: 2 12)
…
Answer: (6 - 4) * (4 + 8) = 24
4 4 6 8
Try a path (a pair of two numbers), see if the remaining numbers can possibly reach the goal 24. If not, backtrack and attempt another
Thought 1:
4 + 8 = 12 (left: 4 6 12)
Thought 2:
6 - 4 = 2 (left: 2 12)
…
Answer: (6 - 4) * (4 + 8) = 24

Figure 1: **Overview of all prompting methods.** From left to right, **Input-Output** Prompting, **CoT** Prompting (Wei et al. 2023) (baseline), original **Multi-Step ToT** Prompting (Yao et al. 2023) (replication), and our **One-Step ToT** Prompting (extension).

Related Work

Existing work has been done to induce multi-step and logical reasoning in LLMs in order to improve their performance on complex tasks.

Chain-of-Thought (CoT) **CoT** was one of the first attempts to unlock reasoning in LLMs through solely prompting instead of utilizing methods of finetuning. Wei et al. 2023 introduced a thought process to an LLM model by dividing an input task into several small, intermediate tasks, subsequently improving LLM performance in tasks like arithmetic, common sense, and symbolic reasoning.

Tree-of-Thoughts (ToT) CoT prompting results in a linear thought process, which may be insufficient for arithmetic tasks that may benefit from forward exploration or backtracking, where specific thought processes may be pruned, or multiple potential solutions should be explored. Therefore, **ToT** (Yao et al. 2023) was developed to improve performance on such tasks. ToT allows for the exploration of multiple potential thought processes by generating different branches of intermediate steps. This method has been shown to improve LLM reasoning significantly in tasks requiring creative exploration, such as Game of 24 and Crossword Puzzles. In the rest of the paper, we often refer to this original ToT prompting as the **Multi-Step ToT** to differentiate from our proposed One-Step ToT.

Knowledge Distillation Magister et al. 2023 introduced a CoT knowledge distillation framework to transfer reasoning capabilities from LLMs (Teacher) to a smaller model (Student). Their approach significantly improved performance on arithmetic, commonsense, and symbolic reasoning tasks by fine-tuning smaller models on CoT outputs generated by larger Teacher models. This work demonstrates the effectiveness of leveraging CoT knowledge distillation to enhance reasoning in smaller models. Our project aims to use knowledge distillation to transfer more powerful and complex ToT reasoning to an SLM to improve SLM performance further.

Background

Game of 24 Dataset Game of 24 is a mathematical reasoning task where the model is given four numbers and must combine these numbers with either addition, subtraction, multiplication, or division to output an expression that yields 24 as the result and that uses each number only once. For example, the model could be given the numbers 1, 1, 4, and 6, which could be solved by doing $4 \times 6 + 1 - 1 = 24$. We choose the Game of 24 dataset used by Yao et al. 2023, which contains 1,362 puzzles, as our benchmark for evaluating language models' reasoning skills because this challenging task demands complex explorations and backtracking.

Input:	**Correct Output:**
Use numbers and basic arithmetic operations ($+$, $-$, $*$, $/$) to obtain 24. Each step, you are only allowed to choose two of the remaining numbers to obtain a new number. Input: 2 9 10 12 Steps: 12 * 2 = 24 (left: 9 10 24) 10 - 9 = 1 (left: 1 24) 24 * 1 = 24 (left: 24) Answer: (12 * 2) * (10 - 9) = 24 {more_in_context_demonstrations}. Input: **4 4 6 8**	Steps: 4 + 8 = 12 (left: 4 6 12) 6 - 4 = 2 (left: 2 12) 2 * 12 = 24 (left: 24) Answer: (6 - 4) * (4 + 8) = 24

Table 1: **An example of CoT Question-Answering for Game of 24 puzzles.** In addition to the puzzle, we have in-context demonstrations encouraging the language model to output intermediate thoughts.

Chain-of-Thought Details CoT is a prompting framework that can be used on LLMs to induce higher levels of reasoning over naive Input-Output (IO) prompting. In IO, the user will only provide the input to describe the task and expects the final answer. In contrast, as seen in Figure 1, CoT prompting asks the model to provide intermediary steps, dividing the overall task into a linear progression of several smaller tasks. Refer to **Table 1** as an example of using CoT prompting on a Game of 24 puzzle.

The number of examples given to the model in the input is referred to k-shot prompting where k describes the number of demonstrations the model is given. In Yao et al. 2023, 5-shot prompting was used, meaning 5 examples were given of Game of 24 puzzles being solved using intermediary steps before the task was given to the model.

Tree-of-Thoughts Details ToT, also referred to as the Multi-Step ToT, represents the reasoning process as a search through a tree of potential solution paths. As illustrated in Figure 1, ToT explores multiple branches of reasoning, allowing models to consider and evaluate various intermediate states and decisions. Each node in the tree represents a partial solution or intermediate reasoning state, while edges signify the steps or transitions between states. ToT uses tree search algorithms such as Breadth-First Search (BFS) or Depth-First Search (DFS) to systematically explore and evaluate paths, scoring them based on a utility function to identify the best solution. This structured approach enables ToT to excel in complex, multi-step reasoning tasks, such as mathematical problem-solving, planning, and decision-making, where multiple solution paths must be considered. In our experiment, we use BFS, and we set a breadth parameter b, which means we keep b candidates in each step.

Methodology, Results, and Discussion

In this section, we replicate the effectiveness of CoT and the original Multi-Step ToT in solving complicated Game of 24 problems and add adaptations, such as developing our own automatic checker and the use of GPT-4o instead of the original GPT-4. Results revealed Multi-Step ToT's superior accuracy, particularly with increased candidate breadth b, achieving up to 82% success rate.

Method	Success
IO prompt (Yao et al. 2023)	7.3%
CoT-SC ($k = 100$) (Yao et al. 2023)	9%
Replicated CoT prompt ($k = 1$)	7%
Replicated ToT ($b = 1$)	4%
Replicated ToT ($b = 3$)	68%
Replicated ToT ($b = 5$)	**82%**
One-Step ToT ($k = 1$)	**19%**

Table 2: **Game of 24 Results.** Replicated ToT (Multi-Step ToT) achieves the best performance as we keep more candidates at each step. Our One-Step ToT, a more efficient prompting method that does not require iterative prompting, beats CoT and ToT at $b = 1$.

Test Set

We utilized the Game of 24 dataset. Following Yao et al. 2023, we used 100 puzzles indexed from 901 to 1,000 as our test set, as these were identified as relatively difficult and served as a robust benchmark for evaluating complex problem-solving capabilities.

CoT Replication Methods

To replicate CoT on the Game of 24, we use the dataset and prompts in Yao et al. 2023's codebase. We also developed our own checker that automatically checks if the model's answer is correct, given that there can be multiple solutions to a Game of 24 puzzle.

We use the CoT five-shot prompt in (Yao et al. 2023) as input for GPT-4o to solve the test set. An **automated checker** was implemented to evaluate the outputs for correctness. This checker flagged outputs as failed if they exhibited hallucinations, incorrect arithmetic, or invalid solutions (e.g., failure to use all input numbers exactly once). Additionally, it identified the specific step for failed cases where the solution diverged from reaching the target value of 24 or contained errors. We used this checker to evaluate CoT's success rate and failure rates at each of the four intermediary CoT steps needed to solve each puzzle correctly.

Due to computational, budget, and time constraints, CoT-SC (Wang et al. 2022) and CoT using the "best-of-k" approach (where puzzles are evaluated using CoT k times and the best result is chosen) were not replicated.

Multi-Step ToT Replication Methods

We replicate the accuracy of Game of 24 by using the Multi-Step ToT in Yao et al. 2023. We used BFS as our search algorithm. At each tree node, the language model (LM) generated multiple thought candidates for the next step. The breadth parameter b determines the number of top candidates retained for further exploration. We tested values of b, including $b = 1$ and $b = 5$, and compared the results. The deliberate BFS approach also incorporated a valuation step, where the LM classified each thought as "sure," "maybe," or "impossible" based on its potential to lead to the solution, thus enabling the pruning of unproductive branches. We employ the same checker to evaluate the success rates.

Models and Parameters

The experimental setup closely followed Yao et al. 2023, with similar API parameters such as `temperature`. However, we used GPT-4o (OpenAI 2024b) instead of GPT-4 (OpenAI 2024a) to streamline the output into a structured `JSON` format, improving the efficacy of our checker. This substitution also enabled us to evaluate whether advancements in the model over time led to improved solutions, which preliminary results suggest they did.

Results

CoT CoT achieved an accuracy of **7%** on the Game of 24 tasks using $k = 1$ (i.e. only running each puzzle once and checking whether the response output is correct).

Multi-Step ToT We tested the accuracy Multi-Step ToT using BFS with $b = 1$, $b = 3$, and $b = 5$. The accuracy increases with the number of nodes visited, as shown in Figure 2. We achieved an accuracy of **4%** when $b = 1$, **68%** when $b = 3$, and **82%** when $b = 5$. These results closely align with the findings in Yao et al. 2023, as the more candidates we keep at each step, the more accurate the final performance.

GPT-4o Improves over GPT-4 Using CoT on GPT-4o, we achieved 7% accuracy instead of 4% for GPT-4. Using ToT on GPT-4o at $b = 5$, we achieved 82% success rate, higher than 74% in GPT-4 reported in Yao et al. 2023.

Discussion

Our replication confirmed that Multi-Step ToT delivers more accurate and robust responses than CoT for solving the Game of 24, and GPT-4o achieves higher success rates than GPT-4 under the same prompting methods.

The BFS breadth parameter b significantly impacts ToT's performance. For $b = 1$, the model retains fewer candidates, leading to faster computations but lower accuracy. On the other hand, $b = 5$ increases the accuracy by maintaining a more diverse set of candidate thoughts, albeit at a higher computational cost. We also concluded that GPT-4o outputs demonstrated improved reasoning and alignment with solution constraints over GPT-4, likely benefiting from more refined training data or architecture updates.

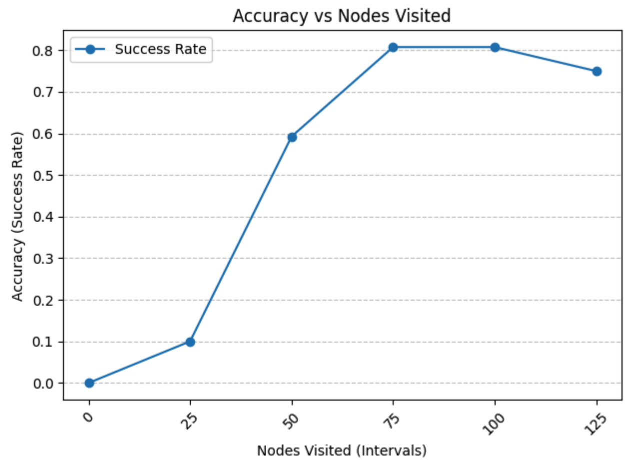

Figure 2: **Success rate with node visited using Multi-Step ToT framework.** The accuracy increases significantly with the number of nodes visited.

As seen in Figure 3, our error analysis revealed that CoT outputs often failed at the initial steps of reasoning, consistent with the challenges of left-to-right decoding. ToT's valuation mechanism mitigated this issue by filtering unpromising branches early in the reasoning process, resulting in significantly higher success rates.

Extensions

In this section, we propose adaptations to the Multi-Step ToT framework, focusing on the streamlined One-Step ToT method. By integrating ToT principles into a single system prompt, One-Step ToT simplifies computations by eliminating iterative prompting while achieving improved accuracy over CoT. We also present a knowledge distillation pipeline, showcasing how fine-tuning a smaller model enables it to rival the performance of a LLM, offering a more efficient alternative.

One-Step ToT

Methods As seen in Table 3, for One-Step ToT, we adopt similar prompting as the CoT but added a **system prompt** that integrates ToT, in addition to the in-context demo that provides several examples. The system prompt instructs the model to consider possible operations for pairs of numbers and try different paths until it is able to reach an answer of 24. The complete prompt, which comprises of the in-context demo and the system prompt, additionally instructs the model to solve a specific puzzle and structure its outputs in a standard format to facilitate easier parsing. Thus, unlike the original Multi-Step ToT, there is no need for multiple input prompts. Following CoT and ToT replications, we used GPT-4o as our model.

Input:	**Correct Output:**
Use numbers and basic arithmetic operations $(+, -, *, /)$ to obtain 24. Each step, you are only allowed to choose two of the remaining numbers to obtain a new number. **Step 1**: Start by considering possible operations for each pair of numbers. **Step 2**: Try a path (a pair of two numbers), see if the remaining numbers can possibly reach the goal 24. If not, backtrack and attempt another. **Step 3**: Branch out to try different orders of operations and combinations, evaluating each outcome. **Step 4**: If one path doesn't lead to a solution, backtrack and try alternative operations. {in_context_demonstrations}. Solve the following puzzle: **4 4 6 8**.	Steps: $4 + 8 = 12$ (left: 4, 6, 12) $6 - 4 = 2$ (left: 2, 12) $2 * 12 = 24$ (left: 24) Answer: $(6 - 4) * (4 + 8) = 24$

Table 3: **An example of One-Step ToT Question-Answering for Game of 24 puzzles.** One-Step ToT is similar to CoT in that it only requires one prompt instead of iterative prompts for a single puzzle. However, One-Step ToT adopts system prompts that encourages trying different paths, branching out to different combinations, and backtracking to a previous operation.

Failure Cases We conducted a failure case analysis on our One-Step ToT compared with CoT and found that not only is One-Step ToT more accurate, but also One-Step ToT is less likely to fail at the first two steps.

Our testing methodology for One-Step ToT was the same as that of the CoT and ToT testing methods. Upon obtaining a JSON file as output from the model, we parsed the output file using the same automated "checker" as that used before. The checker evaluated puzzles 901-1000, flagging any incorrect outputs, keeping track of which line had caused the failure, and noting which condition(s) had failed.

Performance for One-Step ToT slightly differed from that seen on CoT and ToT, though there were also many similarities. As seen in Figure 3, like CoT, the majority of failures in One-Step ToT were on the first step. These failures consisted of thought processes where the model used any of the numbers more than once, used any numbers not given in the original puzzle, or evaluated the expression incorrectly. The fewest failures occurred at Step 2, and Steps 3 and 4 had almost identical amounts of failures in both CoT and One-Step ToT. However, the difference was that, though most failures were at Step 1, there were fewer failures at Step 1 in One-Step ToT than compared to the number seen in CoT. Conversely, as expected, One-Step ToT had more correct examples than CoT; some examples that failed at Step 1 in CoT did not fail with One-Step ToT.

Results One Step ToT achieved an accuracy of **19%**, compared to CoT, which achieved an accuracy of 7%, which is almost triple the accuracy of CoT. We demonstrate how ToT's valuation mechanism of pruning unpromising branches early improves accuracy.

Knowledge Distillation with One-Step ToT

Synthesized Dataset We created and used a synthesized dataset containing correct ToT-style LLM responses to fine-tune an SLM. We first run the LLM, GPT-4o, on the entire Game of 24 dataset of 1,362 puzzles. We then run the checker to extract the correct responses as the final synthesized dataset for fine-tuning the SLM. The number of correct responses that were used is 144, which is 10.57% of the original dataset.

Finally, we divide this synthesized set into two subsets: a training set of 129 puzzles and a validation set of 15 puzzles. Following (Yao et al. 2023) and previous sections, we use the testing set of puzzles indexed from 901 to 1,000 in the original Game of 24 datasets, excluding those already included in the synthesized dataset, to evaluate the SLM's performance before and after our knowledge distillation pipeline.

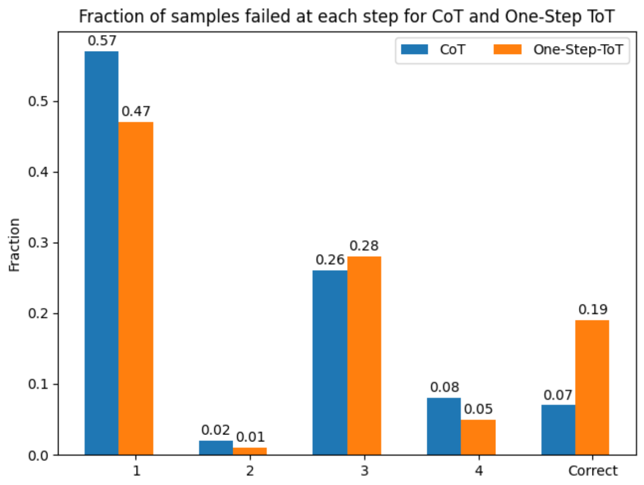

Figure 3: **Comparison of our One-Step ToT and CoT in the fraction of samples failed at each step.** Not only does One-Step ToT have more correct responses, but it also has fewer failures at the first two steps than CoT.

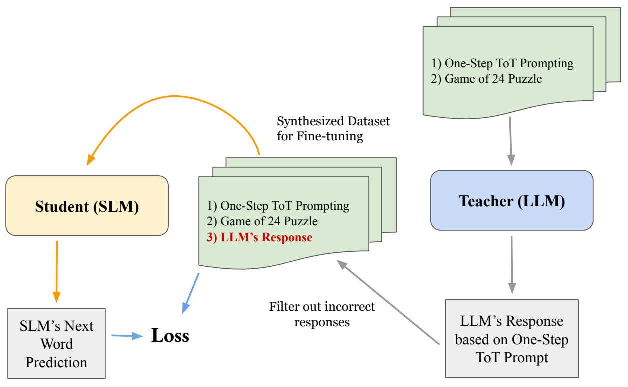
1) One-Step ToT Prompting
2) Game of 24 Puzzle
Synthesized Dataset for Fine-tuning
Student (SLM)
1) One-Step ToT Prompting
2) Game of 24 Puzzle
3) LLM's Response
Teacher (LLM)
Filter out incorrect responses
SLM's Next Word Prediction
Loss
LLM's Response based on One-Step ToT Prompt

Figure 4: **Our proposed knowledge distillation pipeline.** We synthesize a new dataset using the proposed One-Step ToT on an LLM and then fine-tune the SLM to emulate the LLM's responses.

Model We use a small open source language model, SmolLM-360M (Allal et al. 2024). Trained on a high-quality dataset, SmolLM-Corpus (Ben Allal et al. 2024), the model provides a balance between performance and computational efficiency. It has a context length of 2,048 tokens.

Hyperparameters We use Optuna (Akiba et al. 2019) to find the best set of hyperparameters that achieve the highest success rate on the validation set over 20 trials. The resulting hyperparameters are a batch size of 4 and the AdamW optimizer (Loshchilov and Hutter 2019) with a learning rate of 3.17×10^{-5}, and a weight decay of 0.06.

Loss Function Following Radford and Narasimhan 2018, our fine-tuning process leverages a causal language modeling objective to optimize the SLM (Student Model)'s ability to predict the next token in the synthesized target sequence generated by an LLM (Teacher Model), given the preceding tokens (as shown in Figure 4). Formally, the loss function is defined as:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} \log P(y_i \mid x_{1:i-1}; \theta) \quad (1)$$

where $x = \{x_1, x_2, \ldots, x_{i-1}\}$ represents the input tokens, $y = \{y_1, y_2, \ldots, y_N\}$ denotes the target tokens, and θ are the model parameters. This objective enables the model to learn the conditional probability distribution on the token sequences.

Fine-tuning We fine-tune the model on a Nvidia A100 GPU for three epochs. Due to the relatively small size of our training set ($n = 144$), the training loss converged rather quickly, as shown in Figure 5 in the Appendix. To save GPU RAM, we used gradient checkpointing (Chen et al. 2016). Instead of storing all intermediate activations in memory, the model recomputes them on-the-fly during backpropagation.

Results As seen in Table 4, the performance of SLM improves significantly from **1% to 9%** after being fine-tuned on the synthesized dataset. Moreover, its performance exceeded that of GPT-4 and GPT-4o with CoT prompting, when the latter models have vastly larger number of parameters. As expected, the performance is weaker than its teacher, GPT-4o, when prompted with One-Step ToT (19%). However, it should be noted that the Game of 24 dataset ranks the puzzles in ascending difficulty. While our test set, puzzles indexed from 901 to 1,000, sits on the more difficult end of the entire dataset, the training set was collected over GPT-4o's correct responses on the entire dataset. We argue that if we split the training and testing set more randomly, the performance gap might be further reduced.

Model with Prompt Method	Success
GPT-4 with IO prompt (Yao et al. 2023)	7.3%
GPT-4 with CoT-SC ($k = 100$) (Yao et al. 2023)	9%
GPT-4 with CoT ($k = 1$) (Yao et al. 2023)	4%
GPT-4o with Replicated CoT ($k = 1$)	7%
GPT-4o with Replicated ToT ($b = 1$)	4%
Original SmolLM with One-Step ToT	1%
Our **fine-tuned SmolLM** with One-Step ToT	**9%**

Table 4: **Success rate of different models prompted using different methods.** After our proposed knowledge distillation pipeline, the SmolLM with only 360M parameters rivals GPT-4o and GPT-4 that have trillions of parameters.

Conclusions

We introduced the One-Step Tree-of-Thoughts framework combined with knowledge distillation to transfer reasoning capabilities from Large Language Models (LLMs) to Small Language Models (SLMs). Our method simplifies the Multi-Step ToT framework into a single structured prompt and fine-tunes SLMs using a synthesized dataset generated by prompting an LLM. Experimental results on the Game of 24 benchmark demonstrated that this approach enables SLMs to achieve competitive reasoning performance while maintaining computational efficiency.

Future research could explore extending One-Step ToT to a wider range of tasks, including creative problem-solving. Parameter-efficient fine-tuning methods (Xu et al. 2023) can also be used to further improve our knowledge distillation pipeline. Additionally, adapting this framework to multimodal tasks involving both textual and visual inputs (Zhang et al. 2024) could expand its applicability.

Societal Impact

We demonstrated the use of knowledge distillation to equip SLMs with reasoning capabilities, enabling them to handle complex tasks traditionally dominated by LLMs. By achieving comparable performance with significantly reduced computational demands, SLMs offer much faster inference times and are environmentally friendly, addressing the pressing need for sustainable AI solutions (Wang et al. 2024). This advancement sets the stage for more accessible and resource-efficient AI systems.

Individual Contributions

All authors contributed equally to this work.

Zichen Zhang: Proposed and implemented the prompting for One-Step Tree-of-Thoughts; implemented the knowledge distillation pipeline; evaluated the original SmolLM-360M and the fine-tuned one on the Game of 24 test set and recorded the success rates; created Figure 1, Figure 4, Table 3, Table 1, Table 4 and Table 2 in the paper; wrote part of the introduction, related works, extensions, conclusions and appendix sections in this paper.

Shangjun Meng: Revised the original ToT codebase for compatibility with latest OpenAI python module; replicated ToT results with Zimo, generated/synthesized training data for finetuning of the SLM with OpenAI API and scripting; optimized hyperparameters for finetuning, implemented and ran actual finetuning on Google Colab and produced Figure 5; wrote part of Extensions (Knowledge Distillation) of the current paper.

Adi Mahesh: Wrote the main draft for the methodology, results, and discussion section in this paper, along with helping to code for the CoT/ToT results for the replication section by identifying correct answers for each puzzle.

Samuel Fang: Wrote part of the introduction, background, and related work sections for the paper; helped code the CoT replication section and verification of CoT performance compared to the original paper.

Zimo Si: Wrote ToT replication, ToT result, part of background and abstract; coded the ToT replication section and generated the result of ToT performance; created Figure 2 in the paper.

Anurag Renduchintala: Implemented the testing framework (the automated ”checker”) that evaluated GPT-4o's thought process for a given puzzle, and used it to test the model's responses for One-Step ToT. Wrote the extensions section of the paper, describing the procedure and results obtained, along with a figure that compares CoT and our One-Step ToT framework.

Appendix

Implementation Details

Our datasets and implementation details are publicly available at https://github.com/zichenzhang04/slm-tot.

Fine-tuned SmolLM-360M Weights

The weights of our SmolLM model after knowledge distillation using LLM synthesized Game of 24 dataset is publicly available at https://drive.google.com/file/d/11jJ_bEfa7eoDcQBsVY6ydW7kghcGQPLR/view?usp=sharing.

Knowledge Distillation Details

As seen in Figure 5, we stopped the training at step $= 99$. The final training loss is 3.50.

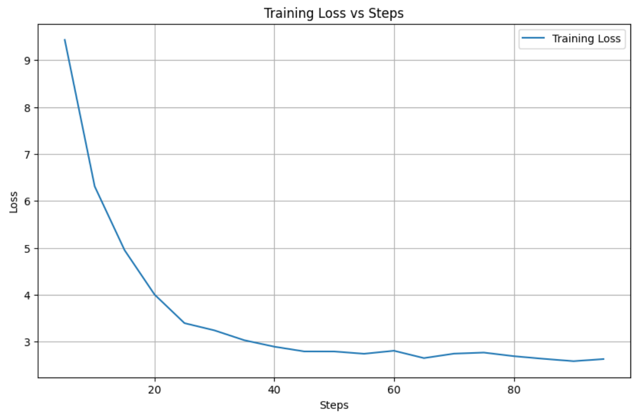

Figure 5: **Training loss with the number of steps.**

References

Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; and Koyama, M. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Allal, L. B.; Lozhkov, A.; Bakouch, E.; von Werra, L.; and Wolf, T. 2024. SmolLM - blazingly fast and remarkably powerful.

Ben Allal, L.; Lozhkov, A.; Penedo, G.; Wolf, T.; and von Werra, L. 2024. SmolLM-Corpus.

Bender, E. M.; Gebru, T.; McMillan-Major, A.; and Shmitchell, S. 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, 610–623. New York, NY, USA: Association for Computing Machinery. ISBN 9781450383097.

Chen, T.; Xu, B.; Zhang, C.; and Guestrin, C. 2016. Training Deep Nets with Sublinear Memory Cost. arXiv:1604.06174.

Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. arXiv:1711.05101.

Magister, L. C.; Mallinson, J.; Adamek, J.; Malmi, E.; and Severyn, A. 2023. Teaching Small Language Models to Reason. arXiv:2212.08410.

OpenAI. 2024a. GPT-4 Technical Report. arXiv:2303.08774.

OpenAI. 2024b. GPT-4o System Card. arXiv:2410.21276.

Radford, A.; and Narasimhan, K. 2018. Improving Language Understanding by Generative Pre-Training.

Wang, F.; Zhang, Z.; Zhang, X.; Wu, Z.; Mo, T.; Lu, Q.; Wang, W.; Li, R.; Xu, J.; Tang, X.; He, Q.; Ma, Y.; Huang, M.; and Wang, S. 2024. A Comprehensive Survey of Small Language Models in the Era of Large Language Models: Techniques, Enhancements, Applications, Collaboration with LLMs, and Trustworthiness. arXiv:2411.03350.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E. H.; and Zhou, D. 2022. Self-Consistency Improves Chain of Thought Reasoning in Language Models. *ArXiv*, abs/2203.11171.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; and Zhou, D. 2023. Chain-of-

Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903.

Xu, L.; Xie, H.; Qin, S.-Z. J.; Tao, X.; and Wang, F. L. 2023. Parameter-Efficient Fine-Tuning Methods for Pre-trained Language Models: A Critical Review and Assessment. arXiv:2312.12148.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T. L.; Cao, Y.; and Narasimhan, K. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. arXiv:2305.10601.

Zhang, Z.; Zhang, A.; Li, M.; Zhao, H.; Karypis, G.; and Smola, A. 2024. Multimodal Chain-of-Thought Reasoning in Language Models. arXiv:2302.00923.

Zhao, W. X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; Du, Y.; Yang, C.; Chen, Y.; Chen, Z.; Jiang, J.; Ren, R.; Li, Y.; Tang, X.; Liu, Z.; Liu, P.; Nie, J.-Y.; and Wen, J.-R. 2023. A Survey of Large Language Models. arXiv:2303.18223.