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

Anurag Renduchintala[1], Aditya 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) is 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 demonstrates 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 replicate 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 propose One-Step ToT, a simplified prompting framework that integrates ToT reasoning into a single structured prompt, and prove its effectiveness over naive CoT. We demonstrate 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.

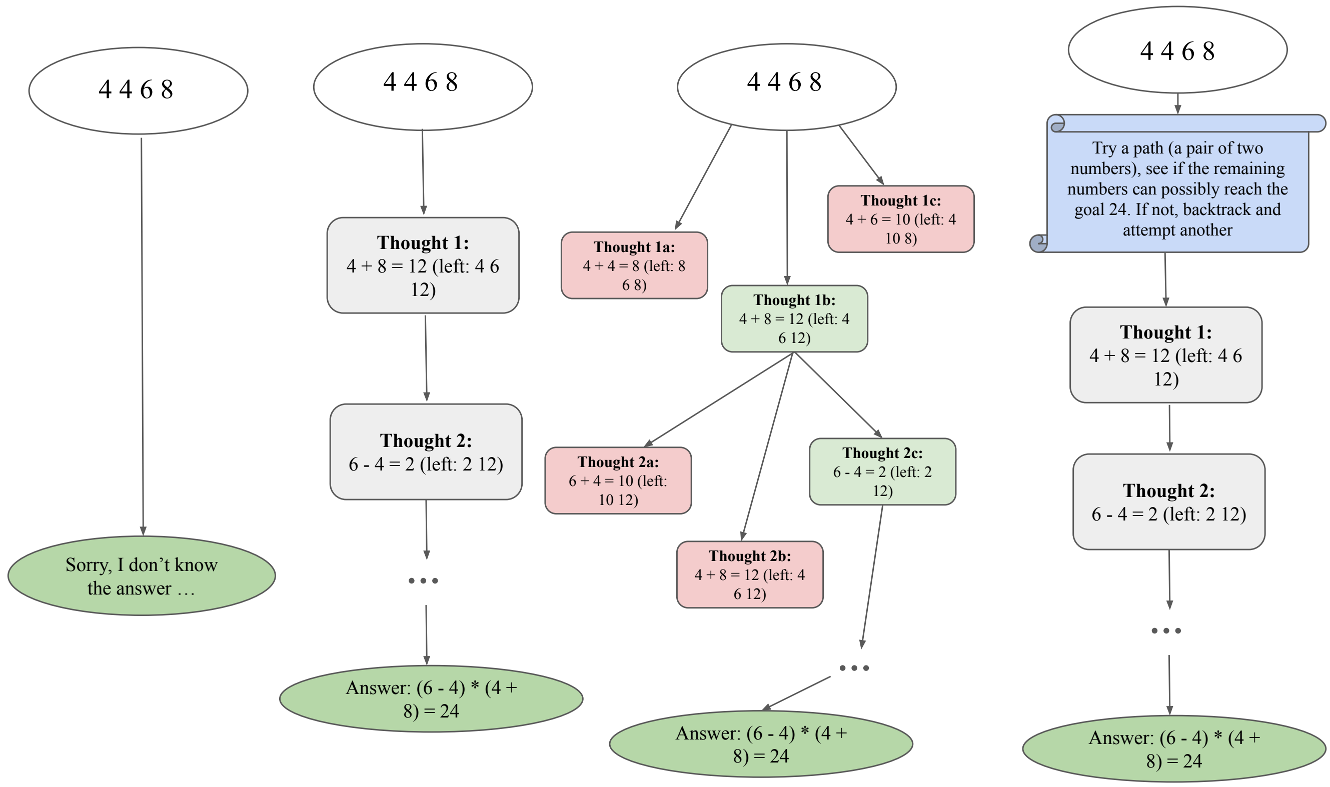

Figure 1: **Overview of all prompting methods.** From left to right, **Input-Output** Prompting, **CoT** Prompting (Wei et al. 2023) (baseline), original **Multi-Step ToT** Prompting (Yao et al. 2023) (replication), and our **One-Step ToT** Prompting (extension).

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** is one of the first attempts to unlock reasoning in LLMs through solely prompting instead of utilizing methods of finetuning. Wei et al. 2023 introduce 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) is 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 introduces a CoT knowledge distillation framework to transfer reasoning capabilities from LLMs (Teacher) to a smaller model (Student). Their approach significantly improves 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 is given the numbers 1, 1, 4, and 6, which could be solved by $4 \times 6 + 1 - 1 = 24$. 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 {`four more 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 five demonstrations (5-shot prompting) 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. The number of demonstrations given to the model in the input is referred to as k-shot prompting, where k describes the number of demonstrations the model is given. In Yao et al. 2023, 5-shot prompting is used, meaning five demonstrations of the Game of 24 solutions that contain intermediary thought processes are presented before the actual task is given to the model. Refer to **Table 1** as an example of using 5-shot CoT prompting to solve the puzzle.

Tree-of-Thoughts Details ToT (also referred to as Multi-Step ToT) is a hybrid prompting and algorithmic framework that represents the reasoning process as a search through a tree of potential solution paths. As illustrated in Figure 1, at each intermediate step, ToT prompts the LLM to generate several potential thought branches, resulting in a tree of potential thought paths where each tree node represents an intermediate reasoning state. At each step, a separate evaluator model scores the likelihood of each thought branch succeeding (from Impossible to Certain) based on the current reasoning state; a breadth parameter b indicates the number of most-likely-to-succeed thoughts to keep at each step. ToT uses tree search algorithms such as Breadth-First Search (BFS) or Depth-First Search (DFS) to explore and evaluate thought paths, scoring them based on a utility function to identify the best solution. ToT excels in complex, multi-step reasoning tasks, such as mathematical problem solving, planning, and decision-making, where multiple solution paths must be considered.

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 reveal Multi-Step ToT's superior accuracy, particularly with increased candidate breadth b, achieving up to 82% success rate.

Experimental Setup

The experimental setup closely follows 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. This substitution also enables us to evaluate whether advancements in the model over time led to improved solutions, which our results suggest they did.

Test Set

We utilize the Game of 24 dataset. Following Yao et al. 2023, we use 100 puzzles indexed from 901 to 1,000 as our test set, as these are identified as relatively difficult and serve 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 develop 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 5-shot CoT prompting in Yao et al. 2023 as input for GPT-4o to solve the test set. We implement an **automated checker** to evaluate the outputs for correctness. This checker flags outputs as failed if they exhibit hallucinations, incorrect arithmetic, or invalid solutions (e.g., failure to use all input numbers exactly once). Additionally, it identifies the specific steps for failed cases where the solution diverged from reaching the target value of 24 or contained errors. We use 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) are not replicated.

Prompt 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: **Success rates of different prompting methods.** 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 $b = 1$.

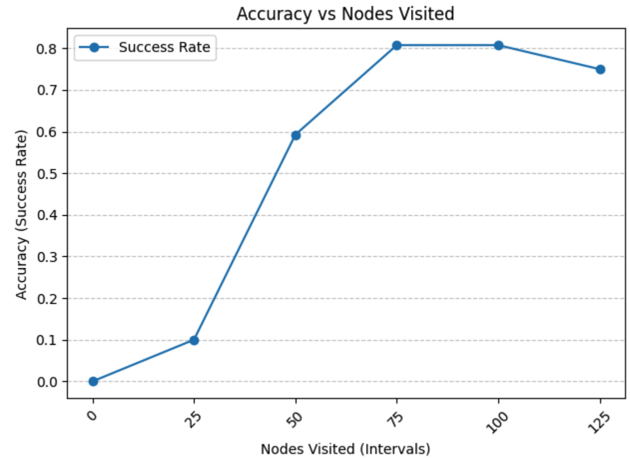

Figure 2: **Success rate with node visited using Multi-Step ToT framework.** The accuracy increases significantly with the number of nodes visited.

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 use BFS as our search algorithm. At each tree node, the language model (LM) generates multiple thought candidates for the next step. The breadth parameter b determines the number of top candidates retained for further exploration at each step. We test different values of b, including 1, 3 and 5, and compare the results. The deliberate BFS approach also incorporates a valuation step, where the LM classifies 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.

Results

CoT As seen in Table 2, CoT achieves 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 test the accuracy of 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. As seen in Table 2, ToT achieves higher accuracy as b becomes higher, closely aligning 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 achieve 7% accuracy, higher than 4% for GPT-4. Using ToT on GPT-4o at $b = 5$, we achieve 82% success rate, higher than 74% for GPT-4 reported in Yao et al. 2023.

Discussion

Our replication confirms 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 conclude that GPT-4o outputs demonstrate improved reasoning and alignment with solution constraints over GPT-4, likely benefiting from more refined training data or architecture updates.

As seen in Figure 3, our error analysis reveals that CoT outputs often fail at the initial steps of reasoning, consistent with the challenges of left-to-right decoding. ToT’s valuation mechanism mitigates 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 a generalizable One-Step ToT method suited for knowledge distillation. By integrating ToT principles into a single prompt, One-Step ToT simplifies computations by eliminating customized iterative prompting that makes knowledge distillation unrealistic while achieving improved accuracy over CoT. Using One-Step ToT, we present a knowledge distillation pipeline, showcasing how fine-tuning an SLM enables it to rival the performance of an LLM, offering a more efficient alternative.

One-Step ToT

Methods As seen in Table 3, for One-Step ToT, we attempt to retain certain benefits of Multi-Step ToT (e.g. multi-thought exploration and backtracking) over CoT, but replace the original Multi-Step ToT framework described in Yao et al. 2023 with a **system prompt** in addition to the in-context demo that provides several examples. This approach uses a pure LLM prompting approach like CoT. 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

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.

comprises 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 Multi-Step ToT replications, we use GPT-4o as our model.

Failure Cases We conduct a failure case analysis on our One-Step ToT compared with CoT and find 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 is the same as that of the CoT and ToT testing methods. Upon obtaining a `JSON` file as output from the model, we parse the output file using the same automated "checker" as that used before. The checker evaluates puzzles 901-1000, flagging any incorrect outputs, keeping track of which line had caused the failure, and noting which condition(s) have failed.

Performance for One-Step ToT slightly differs from that seen on CoT and ToT, though there are also many similarities. As seen in Figure 3, like CoT, most failures in One-Step ToT are on the first step. These failures consist of thought processes where the model uses any of the numbers more than once, uses any numbers not given in the original puzzle, or evaluates the expression incorrectly. The fewest failures occur at Step 2, and Steps 3 and 4 have almost identical amounts of failures in both CoT and One-Step ToT. However, the difference is that, though most failures are at Step 1, there are fewer failures at Step 1 in One-Step ToT than compared to the number seen in CoT. Conversely, as expected, One-Step ToT has more correct examples than CoT; some examples that fail at Step 1 in CoT do not fail with One-Step ToT.

Results and Limitations As seen in Table 2, Our One-Step ToT achieves an accuracy of **19%**, almost triple the CoT's accuracy. As expected, One-Step ToT, a much-simplified version that only requires one prompt rather than

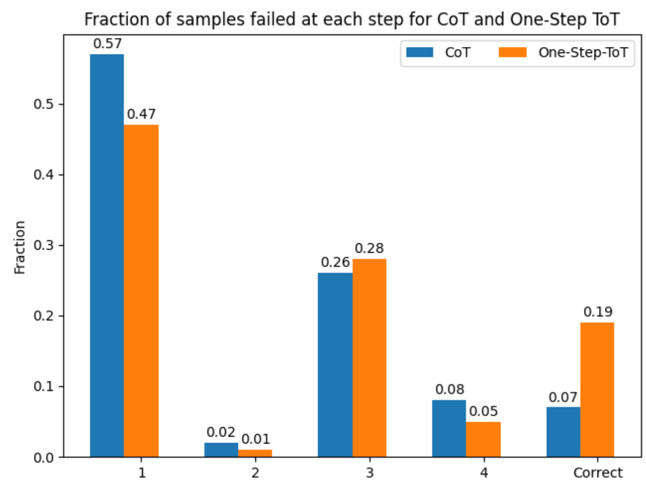

Figure 3: **Comparison of our One-Step ToT and CoT in the fraction of samples failed at each step.** Not only does One-Step ToT have more correct responses, but it also has fewer failures at the first two steps than CoT.

multiple prompts used in Multi-Step ToT, is still worse than the best performance of Multi-Step ToT at $b > 1$, though it's more accurate than that at $b = 1$. This shows that there is still room for improving the One-Step ToT prompt.

Knowledge Distillation with One-Step ToT

Synthesized Dataset We create and use 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 execute the checker to extract the correct responses as the final synthesized dataset for fine-tuning the SLM. The number of correct responses used is 144, which is 10.57% of the original dataset.

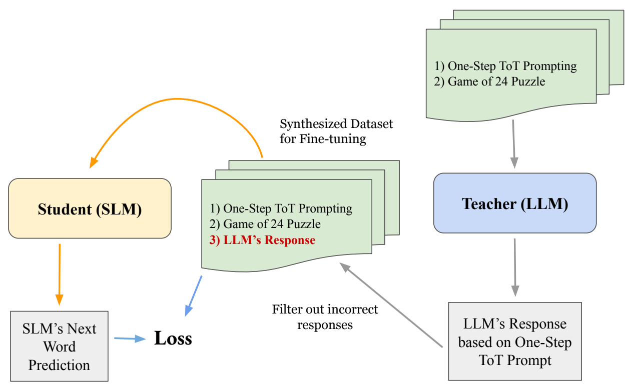

Figure 4: **Our proposed knowledge distillation pipeline.** We synthesize a new dataset using the proposed One-Step ToT on an LLM and then fine-tune the SLM to emulate the LLM's responses.

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.

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:

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

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.

Results We fine-tune the model on an Nvidia A100 GPU for three epochs. 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 exceeds GPT-4 and GPT-4o with CoT prompting, when the latter models have vastly more parameters. As expected, the performance is weaker than its teacher GPT-4o, which has 19% success rate.

In addition, 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 rates 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.

Limitations Despite employing regularization techniques, our fine-tuned model could be overfitting to the Game of 24, leading to reduced performance on other reasoning tasks. However, we hypothesize that if we have enough computing power, we could fine-tune the SLM with a synthesized dataset spanning multiple reasoning benchmarks using our proposed pipeline to increase generalizability.

Conclusion

We introduce 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 distillation pipeline.

Societal Impact

Our extensions demonstrate the potential to equip SLMs with reasoning capabilities that rival LLMs. SLMs, with faster inference times and reduced environmental impact, align with the demand for sustainable AI technologies.

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 Pretrained 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.

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.

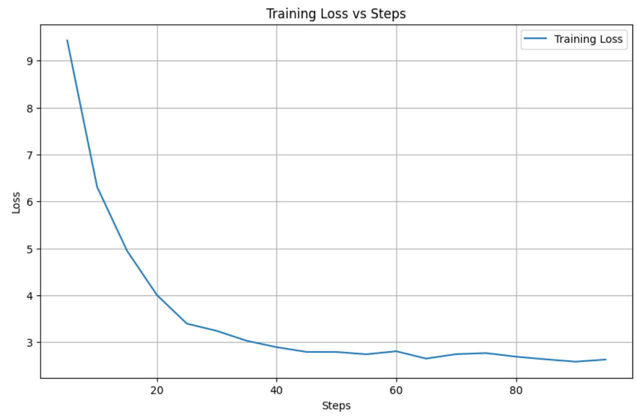

Figure 5: **Training loss of SmolLM-360M Knowledge Distillation with the number of steps.** The loss converges quickly because of the small size of our synthesized dataset, and we stop the fine-tuning early to avoid over-fitting the model to the Game of 24 benchmark.

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.

Knowledge Distillation Details

As seen in Figure 5, we stop the training at the 99th step to avoid over-fitting the model to the Game of 24. Due to the relatively small size of our training set, the training loss converges rather quickly.

To save GPU RAM, we use gradient checkpointing (Chen et al. 2016). Instead of storing all intermediate activations in memory, the model recomputes them on-the-fly during backpropagation.