LLMs Can Plan Faster Only If We Let Them

Anonymous Authors[1]

Abstract

Large language models (LLMs) are making inroads into classical AI problems such as automated planning, yet key shortcomings continue to hamper their integration. Chain-of-Thought (CoT) struggles in complex multi-step reasoning, and Tree-of-Thoughts requires multiple queries that increase computational overhead. Recently, Algorithm-of-Thoughts (AoT) have shown promise using in-context examples, at the cost of significantly longer solutions compared to CoT. Aimed at bridging the solution length gap between CoT and AoT, this paper introduces AoT-O3, which combines supervised finetuning on AoT-style plans with a reinforcement learning (RL) framework designed to reduce solution length. The RL component uses a reward model that favors concise, valid solutions while maintaining planning accuracy. Empirical evaluations indicate that AoT-O3 shortens solution length by up to 80% compared to baseline AoT while maintaining or surpassing prior performance. These findings suggest a promising pathway for more efficient, scalable LLM-based planning.

1. Introduction

Developments in large language models (LLMs) (Vaswani, 2017; Radford, 2018; Radford et al., 2019; Brown et al., 2020; Chowdhery et al., 2023; Zhao et al., 2023, *inter alia*) have led to remarkable advancements in artificial intelligence, particularly in natural language processing. These models, pre-trained on vast corpora of data, have demonstrated impressive capabilities across diverse domains, successfully transferring their knowledge to downstream tasks including code generation (Chen et al., 2021; Li et al., 2022; Jiang et al., 2024), language understanding (Yuan et al., 2022; Katz et al., 2024), instruction following (Ouyang et al., 2022; Bai et al., 2022; Rafailov et al., 2024), optimization (Jin et al., 2024), and general problem-solving (Brown et al., 2020; Nye et al., 2021; Wei et al., 2022; Huang & Chang, 2022). Despite these successes, their performance in complex tasks requiring diverse thought processes, particularly in planning domains (Kambhampati et al., 2024a; Valmeekam et al., 2024), has remained notably limited. This limitation becomes particularly evident in scenarios requiring long-horizon reasoning, where the models must maintain coherence and logical consistency across multiple steps while exploring various solution paths.

The introduction of methods such as Chain-of-Thought (CoT) (Wei et al., 2022), Least-to-Most prompting (L2M) (Zhou et al., 2022), Self-Consistency CoT (CoT-SC) (Wang et al., 2022), and Self-Refine (Madaan et al., 2024) has failed to significantly improve planning capabilities, even with state-of-the-art LLMs. These approaches, while effective in simpler reasoning tasks, struggle with the complexity inherent in planning problems where a single misstep can lead to an unrecoverable state. This limitation has sparked the development of various approaches aimed at diversifying reasoning paths (Long, 2023; Yao et al., 2024; Lei et al., 2023; Yao et al., 2023; Besta et al., 2024). These methods have demonstrated considerable improvements in accuracy compared to CoT by allowing models to explore multiple solution paths simultaneously. However, they face significant challenges: computational expense due to the need for multiple model queries, implementation complexity when applying to new problems, and performance that still falls short of human benchmarks (Sel et al., 2024b; Kambhampati et al., 2024b). The gap between model performance and human capability remains particularly pronounced in domains requiring strategic thinking and adaptive reasoning.

Recent innovations, notably the LLM-Modulo framework (Valmeekam et al., 2024) and Algorithm of Thoughts (AoT) (Sel et al., 2024b; 2025), have shown promising results in approaching human-level performance. LLM-Modulo addresses the limitations of self-feedback by incorporating external verifiers to provide fine-grained feedback to LLMs, effectively creating a closed loop system that can identify and correct errors in real-time. This approach has demonstrated remarkable success in improving model accuracy, but the reliance on external verification tools introduces additional complexity and computational overhead. In contrast, AoT presents a pure LLM solution that eliminates the need for external tools or additional prompting mechanisms. It achieves self-correction by incorporating in-context examples that mirror human thought processes, either expanding upon search nodes or transitioning to more promising nodes that appear likely to reach the goal. This approach lever-

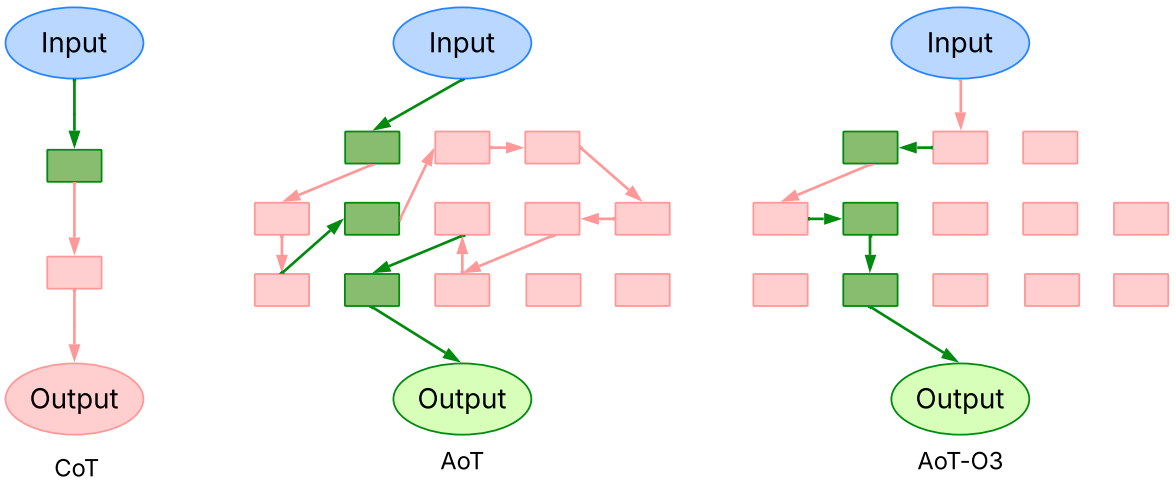

Figure 1. Illustration showing three reasoning strategies with LLMs: Chain of Thought (CoT), Algorithm of Thoughts (AoT), and AoT-O3. Each box represents a thought step, with green indicating promising paths and red showing less promising alternatives. AoT-O3 demonstrates a more streamlined structure while maintaining the key benefits of branching exploration.

ages the model's inherent ability to learn from examples while maintaining the efficiency of single-query solutions. Subsequent research has revealed that these example search processes need not explicitly contain human intuitions; simply augmenting correct CoT-like solutions with random step-by-step solutions enables models to achieve comparable performance (Sel et al., 2025). This finding suggests that the key to improved performance lies not in mimicking human reasoning patterns specifically, but in providing structured examples of successful problem-solving strategies.

While AoT represents a significant advancement in performance requiring only a single query per problem solution, the inclusion of multiple reasoning paths substantially increases the total solution length compared to CoT. This expansion introduces proportional increases in computational requirements and solution time per problem, raising concerns about scalability and efficiency. Excessively long solutions also raise environmental concerns due to increased token usage (Earth.Org, 2024)[1]. Critical questions emerge regarding the necessity and effectiveness of these additional steps: What proportion of these additional steps represents intentional exploration by the model versus mere imitation of training examples? How much of the extended solution length contributes meaningfully to problem-solving versus serving as stylistic overhead? If unnecessary search paths exist primarily to match the style of training examples, can we enable more efficient problem-solving by removing this constraint while maintaining the benefits of diverse reasoning paths? Understanding these aspects is crucial for developing more efficient approaches that maintain the advantages of AoT while reducing computational overhead.

These solution strategies can be analyzed through the lens of human cognitive systems, providing insights into their operational mechanisms and potential improvements. While CoT and L2M align with the analytical *System 2* thinking (Kahneman, 2011), contrasting with reflexive *System 1* decision-making, recent work (Sel et al., 2025) suggests that AoT activates *System 3* thinking (Webb, 2021). This third system, characterized by deliberate contemplation and strategic decision-making in the face of uncertainty, represents a more sophisticated level of cognitive processing. Although we acknowledge this perspective, we propose that current AoT implementations may be imitating rather than truly engaging in *System 3* thinking, pointing to the potential for more efficient and deliberate reasoning path exploration. This distinction between imitation and genuine engagement with different cognitive systems has important implications for the development of more effective problem-solving approaches.

This paper introduces AoT-O3, named after the highest optimization level in C++ programming language compilers, which aims to bridge the solution length gap between CoT and AoT while maintaining or improving performance. Our approach represents a fundamental shift in how LLMs can be guided to solve complex problems more efficiently. We employ reinforcement learning (RL) with carefully designed objective reward models to encourage more concise AoT responses without sacrificing solution quality. The reward models are specifically crafted to balance the competing objectives of solution efficiency and accuracy, ensuring that

[1]Recent projections suggest AI-related power demand could grow substantially in coming years (Agency, 2024). See Appendix A for an estimate of the potential impact.

shorter solutions do not come at the cost of reduced performance. Notably, we pioneer the use of non-LLM reward models that extend beyond simple accuracy metrics, making the fine-tuning process more efficient and memory-friendly. This innovation addresses one of the key limitations of existing approaches that rely heavily on LLM-based reward models, which can introduce significant computational overhead and potential biases as illustrated in Figure 1.

We demonstrate our method's effectiveness across a wide range of open-source LLMs and diverse benchmarks, providing comprehensive evidence of its generalizability and robustness. Our evaluation encompasses various planning domains, from simple sequential tasks to complex strategic problems requiring multiple levels of reasoning. Results show that AoT-O3 achieves an 80% reduction in solution length while improving accuracy, an advantage that becomes *even more critical under tight token constraints*. This remarkable improvement in efficiency does not come at the cost of solution quality; in fact, we observe consistent improvements in accuracy across most benchmarks. The success of AoT-O3 in maintaining or enhancing performance while dramatically reducing computational requirements represents a significant step toward more practical and scalable applications of LLMs in complex problem-solving domains.

2. Related Work

Evolution of LLM Reasoning Approaches. The emergence of large language models has revolutionized approaches to complex problem-solving tasks. Initial work demonstrated that these models, trained on diverse textual data, could tackle various challenges through direct prompting (Brown et al., 2020; Chowdhery et al., 2023). This capability was significantly enhanced through the development of step-by-step reasoning methods that reframe problems as sequential decision processes. Chain-of-Thought prompting (Wei et al., 2022; Kojima et al., 2022) and its variants (Zhang et al., 2022; Nye et al., 2021) marked a crucial advancement in structured reasoning. However, these approaches revealed limitations in inherently sequential domains, particularly planning problems (Long, 2023). Subsequent research introduced more sophisticated frameworks like Tree of Thoughts (Yao et al., 2024), which spawned numerous extensions (Lei et al., 2023; Besta et al., 2024). While these methods improved accuracy by leveraging LLMs as heuristic generators within external search frameworks, they introduced significant computational overhead through multiple API calls. The Algorithm of Thoughts framework (Sel et al., 2024b) addressed these efficiency concerns by demonstrating that carefully crafted in-context examples incorporating search trajectories could achieve comparable or superior performance with a single query. AoT+ (Sel et al., 2025) further improved the ease of use and performance of AoT, although in this paper we will use AoT as the naming for both.

Self-Improvement and Verification in LLMs. The concept of autonomous improvement through self-verification has been extensively explored in LLM research. Initial efforts focused on constitutional training (Bai et al., 2022) and ethical decision-making (Ma et al., 2023; Sel et al., 2024a), demonstrating that models could be guided to evaluate and refine their outputs. This approach has shown promise in specific domains such as code generation (Zelikman et al., 2023; Kim et al., 2024) and question-answering (Madaan et al., 2024; Paul et al., 2023). Recent work has expanded these concepts through recursive self-improvement (Zelikman et al., 2023; 2024). However, significant challenges remain in symbolic reasoning tasks (Valmeekam et al., 2023; Kamoi et al., 2024), where self-verification often fails to identify critical errors. These limitations have sparked debate about the fundamental capabilities of LLMs in autonomous error correction (Kambhampati et al., 2024a), particularly in domains requiring precise logical reasoning.

Reinforcement Learning for LLM Alignment. Reinforcement learning has emerged as a powerful approach for aligning language models with human preferences and improving their capabilities across various tasks. The foundation was laid by InstructGPT (Ouyang et al., 2022), which demonstrated that RL from Human Feedback (RLHF) could effectively tune models to better follow instructions. This was further developed by Constitutional AI (Bai et al., 2022), which showed how recursive reward modeling could instill specific behavioral constraints. Recent work has explored alternative reward mechanisms, with Direct Preference Optimization (DPO) (Rafailov et al., 2024) providing a more computationally efficient alternative to traditional RLHF. The application of RL to improve reasoning capabilities has been particularly noteworthy, with ReAct (Yao et al., 2022) showing how reward signals could encourage more structured exploration of solution spaces.

3. Effect of In-Context Examples in Planning

Before we decide which type of examples to choose for supervised and RL finetuning for our method, we proceed with exploring how the style of in-context demonstrations affect the LLMs when they are prompted with a test problem. Here, we are interested in the differences between prompting and supervised finetuning. First, let's define planning problems of interest.

Classical Planning Problems. (Russell & Norvig, 1995) A *classical planning problem* can be defined as finding a sequence of actions that transitions the system from an initial state I to a goal state G, given:

- A set of possible states S,
- A set of actions A, where each action has:
  - Preconditions that must hold true for the action to be applied,
  - Effects that describe how the action changes the state.

The goal is to compute a plan $\pi = [a_1, a_2, \ldots, a_n]$ such that applying these actions in sequence achieves G starting from I.

3.1. Effect of Random Demonstrations in Finetuning

The role of demonstrations in chain-of-thought prompting was fundamentally reexamined by Min et al. (2022), showing that randomly generated reasoning paths could achieve comparable performance to carefully crafted demonstrations. Their study revealed that the presence of step-by-step reasoning patterns, rather than their actual correctness or optimality, was the key factor in enabling successful problem-solving behavior in language models. This finding suggested that demonstrations primarily serve to elicit a particular problem-solving format from the model, rather than to transmit specific solution strategies. Their work challenged the prevailing assumption that high-quality demonstrations were necessary for effective CoT prompting, indicating that the structural aspects of reasoning demonstrations might be more important than their content.

Method	Accuracy (%)	
	CoT	AoT
Prompting	8	64
Finetuning	3	5

Table 1. Random demonstrations performance of Llama-3.3-70B with greedy-decoding on Game of 24 when prompted in 8-shot setting and finetuned with 900 training and 100 test examples from Yao et al. (2024); Sel et al. (2024b).

This insight has been recently extended to AoT. It's been demonstrated that augmenting correct AoT solutions with random search trajectories achieved performance comparable to carefully designed examples incorporating human intuition (Sel et al., 2025). This shows that the benefits of diverse reasoning paths in AoT persist even when some of the demonstrated search trajectories are suboptimal or irrelevant to the solution. This finding suggests that the key advantage of AoT may lie not in its ability to mimic human-like strategic thinking, but rather in its provision of a structured framework for exploring multiple solution paths, regardless of their individual quality.

However, while random demonstrations may be effective for in-context prompting, their utility appears to diminish significantly in model fine-tuning scenarios. As shown in Table 1, when testing on the Game of 24 benchmark using the Llama 3.3 70B model (Dubey et al., 2024), prompting with random examples maintains comparable performance to gold-standard demonstrations in both CoT and AoT settings. In stark contrast, fine-tuning on random examples leads to a dramatic deterioration in model performance, with accuracy dropping below pre-finetuning levels. This disparity suggests that while models can effectively filter and utilize random demonstrations during inference, incorporating such noise during training may interfere with the model's ability to learn robust reasoning strategies.

3.2. Effect of Solution Length in AoT

To investigate the relationship between demonstration solution length and model performance, we conducted experiments using the OpenAI GPT-4 model (Achiam et al., 2023) on the Game of 24 benchmark (Yao et al., 2024; Sel et al., 2024b). We examined this relationship in both in-context learning (ICL) and fine-tuning scenarios, with particular attention to how different solution length distributions affect model accuracy and efficiency. For our ICL experiments, we constructed three sets of 10-shot demonstrations (per each test example, we select a new 10-shot examples), each following a Gaussian distribution centered around different mean solution lengths: short (10 steps), medium (30 steps), and long (60 steps) with std of 5.0 with a clip at 3.0 since the minimum number of steps in this game is three. These demonstrations maintained consistent solution strategies and reasoning patterns while varying in their degree of elaboration and search path exploration.

As seen in Figure 2, our results indicate a strong positive correlation between solution length and model performance, with longer demonstrations leading to longer solutions produced by GPT-4 with higher accuracy rates, 15%, 31% and 61% with temperature of 0.5. These findings indicate that solution length serves as more than just stylistic variation—it appears to be fundamentally connected to the model's ability to reason effectively about complex problems. The superior performance of longer solutions may be attributed to their more comprehensive exploration of the solution space, allowing the model to better understand and replicate successful problem-solving strategies. These results have important implications for both prompting and fine-tuning strategies, suggesting that artificially constraining solution length may inadvertently limit model performance.

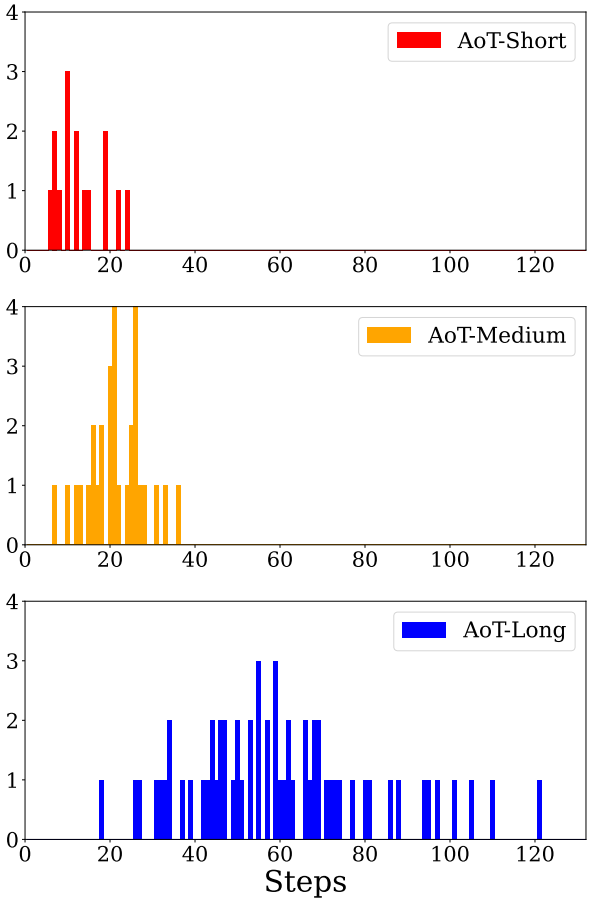

Figure 2. Effect of varying steps for demonstrations to generated solutions in AoT

4. AoT-O3

In a typical RLHF pipeline, we have **SFT stage**, **reward model training stage** and finally **RL training stage**. In our framework, we will incentivize models to generate solutions with less total number of steps while still generating valid solutions. Since both of these objectives are *objective*, we do not necessarily need the second stage which is the reward model training stage, however, we will talk about the reward model in its own subsection nevertheless.

4.1. SFT Stage

The supervised finetuning (SFT) stage aims to train the model to follow AoT-style planning while maintaining solution quality. Similar to how AoT uses carefully crafted in-context examples to guide LLMs' planning process, we create a training dataset that demonstrates effective planning strategies. However, instead of relying on human-authored intuitions in the search trajectories, we utilize random search paths that lead to valid solutions.

For each problem in our training set, we first obtain a valid solution that successfully reaches the goal state. This solution can be obtained through various means, including traditional planning algorithms or even using an LLM. While using an LLM might initially yield a lower percentage of correct solutions compared to curated datasets, this approach offers greater flexibility and scalability. The efficiency trade-off becomes a design choice based on specific requirements and available resources.

Our training examples are structured around the concept of state transitions guided by explicit thinking steps. Each transition follows the format:

```
State: [Current state description]
Thinking/Transition: [Selection of an
action and considering its effects]
Next state: [New state description]
```

The thinking component serves as a crucial bridge between states, providing the model with explicit reasoning that justifies the transition. This structured format helps the model learn the relationship between states, actions, and their consequences. Importantly, any given state in the sequence can serve as a starting point for the next transition, allowing for both forward progression and backtracking when necessary.

An example output from the model could be represented as $y = \{(s_0 \xrightarrow{a_0} s_{00}), (s_0 \xrightarrow{a_1} s_{01}), (s_{00} \xrightarrow{a_{01}} s_{000}), \cdots, (s_{23424} \xrightarrow{a_{23424}} s_{\text{goal}})\}$ or $y = \{T_1, T_2, \cdots, T_{\text{final}}\}$. As seen from this representation, at each step, the model has a choice to start from a previous state and an action to perform on that state. This leads to partial expansion, which is especially effective for memory usage when the number of branches are very large in classical planning problems with programs. In our case, we are limited by the context in two ways: firstly, LLMs have a maximum context window limit that they are pre-trained for, and secondly, it is observed in the literature that LLMs may fail to attend to some parts of the context in large context sizes, especially the middle (Liu et al., 2024). There exists works showing possible heuristics and algorithms that are complete (like A*) such as Partial-Expansion A* (Yoshizumi et al., 2000; Felner et al., 2012).

We generate random but valid exploration paths by starting from either the initial state or intermediate states. At each state, the model considers possible actions and their consequences through the thinking step before transitioning to the next state. This process creates a diverse set of trajectories that may include both promising paths toward the goal and dead-ends that require backtracking.

These exploration trajectories are then combined with segments of successful solution paths. The resulting training examples demonstrate both the exploration of the search

space and eventual convergence to valid solutions. This combination teaches the model to balance exploration with goal-directed behavior.

The SFT objective follows standard language model finetuning approaches, using teacher forcing with a cross-entropy loss:

$$\mathcal{L}(\theta) = -\sum \log p_\theta(y_t|x, y_{<t}) \tag{1}$$

where $x = (s_{\text{start}}, s_{\text{goal}})$ is the input problem specification and y is the target sequence containing the state-thinking-transition triplets.

The training process develops several key capabilities in the model. It learns to systematically explore the search space while maintaining accurate state tracking. The explicit thinking steps help the model develop robust reasoning about state transitions and their consequences. The model also learns to recognize promising solution paths and successfully navigate toward goal states.

This supervised learning phase establishes the foundation for the subsequent RL optimization stage. While the SFT stage focuses on teaching the model the basic structure of planning with thinking steps, the RL stage will introduce rewards to encourage more efficient solutions while maintaining these core planning capabilities.

4.2. Reward Model

While traditional reward models in language model alignment often rely on human feedback or complex learned reward functions, our objective is more straightforward: we with the model to generate solutions that are both correct and efficient. To this end, we develop a simple but effective reward function that balances solution validity with length efficiency. In this section, we will use V for our reward models due to them actually being value models used in RL frameworks. This difference stem from the fact that most RLHF frameworks consider the whole solution as a single action, therefore framing the problem as a single-step RL, in which reward and value models are the same. In our framework, we consider each transition step as an action, leading to our following feedback models being value models. If they were to be thought of as actual reward models, due to their definitions, the language model would try to *hack the reward* by being close to the solution but prolonging the solution to receive more total reward, conflicting with our intention of shorter solutions.

The base reward structure assigns +1 for correct solutions and -1 for incorrect ones, providing a clear signal for solution validity. However, this alone would not encourage the model to find more concise solutions. To address this, we introduce a step count penalty term for correct solutions:

$$V(T_i, y) = \begin{cases} \max(1 - n\alpha, \beta) + \kappa \mathbf{1}_{T_i \in y^*} & \text{if } y \text{ is correct} \\ -1 & \text{if } y \text{ is incorrect} \end{cases} \tag{2}$$

where n is the number of steps in the solution, α is the step penalty factor, β is the cut-off to protect long solutions being given less reward then incorrect solutions, e.g., $\beta = -0.5$, κ controls the incentive if the transition was in the path of the correct solution, and y^* is the correct ***CoT*** solution that can be extracted from a correct solution y. In cases where it might be difficult to extract correct solution trace, we can simply set $\kappa = 0$. We show in our experiments that κ term is not crucial, though helps with faster adaptation and slightly higher performance. This formulation creates a clear trade-off: while the model is primarily incentivized to find correct solutions, it receives higher rewards for doing so with fewer steps.

The choice of α is crucial and depends on both the problem domain and the context window constraints. For a given problem type and context window size w, we aim to set α such that:

$$1 - w\alpha \gtrsim \beta \tag{3}$$

This ensures that a correct solution using the maximum available context length receives a reward slightly higher than an incorrect solution. Such calibration helps to have more fine-grained levels of brevity to rewarded correctly.

Our reward model notably differs from traditional approaches in RLHF by eschewing learned reward models in favor of this objective metric. This design choice offers several advantages: it eliminates the need for reward model training, reduces computational overhead during RL training stage and provides clear, interpretable feedback to the model. Most importantly, it creates a direct optimization pressure for finding shorter valid solutions while maintaining the priority of solution correctness.

Reward model alternatives. The problems we focus on, i.e., planning problems, typically have the trait of verifying the solutions are much easier than producing solutions to them. This assumption also affects the reward function we propose in (2). In cases where we have more compute available but the number of examples are more limited, it is possible to have even finer grained feedback to the model still with the assumption of no human-feedback. Consider the following reward model,

$$V_{\pi_\theta}(T; s_{\text{goal}}) = \mathbb{E}_{y \sim \pi_\theta(\cdot|x=(T', s_{\text{goal}})} \mathbf{1}(\text{solution } y \text{ is correct}), \tag{4}$$

where T' refers to the ending state in transition T and $x = (T', s_{\text{goal}})$ is the new problem where we prompt the model. Here, we would like to attract attention to binary reward is coming from not the solution of the model for the starting

problem, but the newly constructed problem at each state. This enables the model to learn per each transitions whether the whole solution was correct or not and whether T_i is in the path that lead to the correct solution. In reward function (2), we either gave negative feedback for all transitions in case of an incorrect solution, or gave a positive reward for all transitions when the solution was correct. However, the reward function in (2), also makes the distinction whether a transition was indeed important for the solution of the problem by the κ term.

4.3. RL Training Stage

The reinforcement learning stage builds upon the supervised finetuning by optimizing the model for solution efficiency while maintaining correctness. The core objective is to maximize the expected reward under the policy distribution:

$$\mathcal{L}_{RL}(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[\mathbb{E}_{T_i\in y} V(T_i, y)\right]\right] \quad (5)$$

where $\mathcal{D}$ is the dataset of problems, π_θ represents the language model being trained, x represents the input planning problem, and $R(\cdot,\cdot)$ is our length-aware reward function that balances solution correctness with efficiency defined in (2).

This optimization can be accomplished through various RL algorithms developed for language models. Trust Region Policy Optimization (TRPO) (Schulman et al., 2015), Proximal Policy Optimization (PPO) (Schulman et al., 2017; Ziegler et al., 2019; Ouyang et al., 2022) and REINFORCE leave-one-out (RLOO) (Ahmadian et al., 2024) have all demonstrated success in language model alignment tasks. Each algorithm offers different tradeoffs between training stability, computational efficiency, and implementation complexity.

The key distinction of our approach lies in the reward structure rather than the specific RL algorithm choice. Our reward function provides clear, objective feedback about both solution correctness and length efficiency, eliminating the need for learned reward models or complex advantage estimation schemes. This simplification allows us to focus on the core challenge of finding shorter valid solutions without introducing additional training complexity. In our experiments later, we will utilize RLOO.

To maintain stability during RL training, we employ a reference model (the SFT model) to ensure the optimized policy does not deviate too far from learned planning behaviors. This constraint can be implemented through various mechanisms depending on the chosen RL algorithm, such as KL divergence penalties or direct probability ratio clipping. In our setting, the KL divergence has the added benefit of entropy regularization, popular in general RL setting. It is known to help with mode collapse, an unwanted phenomenon that occurs when the model *exploits* without enough *exploration*, leading to suboptimal policies (Williams & Peng, 1991; Ziebart et al., 2008; Mnih, 2016).

The training process samples planning problems from our dataset and allows the model to generate complete solutions, receiving rewards based on both correctness and solution length. This creates a direct optimization pressure toward finding more efficient solutions while maintaining the fundamental planning capabilities established during the SFT stage.

5. Experimental Results

We evaluate AoT-O3 across two challenging planning benchmarks: Game of X and N-Puzzle. Our experiments utilize three open-source models of varying sizes: Gemma2-2B, Llama3-1B, and Llama3-3B. We compare our approach against two baselines: traditional Chain-of-Thought with supervised fine-tuning (CoT-SFT) and Algorithm of Thoughts with supervised fine-tuning (AoT-SFT). Our evaluation focuses on both solution accuracy and efficiency, measured by the number of reasoning steps required to reach a solution.

5.1. Problem Setups

Game of X. We introduce a generalized variant of the Game of 24, which presents a significantly more challenging planning problem. In our version, five numbers are randomly sampled, and players must construct mathematical expressions using arbitrary combinations of addition, subtraction, multiplication, and division operations to reach a target value. This formulation creates a substantially larger search space compared to the traditional Game of 24, requiring more sophisticated planning and arithmetic reasoning capabilities.

N-Puzzle. We utilize the classic 8-puzzle variant of the sliding tile puzzle, where eight numbered tiles are arranged on a 3×3 grid with one empty space. Each puzzle instance is generated by starting from the goal state (tiles arranged in numerical order) and applying a random sequence of 80-120 valid moves to ensure solvability. The objective is to return the tiles to their original ordered configuration by sliding tiles into the empty space. This creates a challenging planning problem with a branching factor of up to 4 at each step and requires careful consideration of move sequences to avoid cycles or dead ends.

Word Ladder. We implement the classic word ladder puzzle, where the objective is to transform one word into another by changing a single letter at a time, with each intermediate step forming a valid English word. Our dataset is constructed using the NLTK base words dataset, focusing on

Problem	Method	Accuracy (%)			Solution Length (steps)		
		Gemma2-2B	Llama3-1B	Llama3-3B	Gemma2-2B	Llama3-1B	Llama3-3B
Game of X	CoT-SFT	32	24	35	4	4	4
	AoT-SFT	63	55	66	33.5	31.9	37.2
	AoT-O3 ($\kappa = 0.0$)	74	71	79	18.3	11.6	15.0
	AoT-O3	**78**	**72**	**82**	16.3	9.7	14.5
N-Puzzle	CoT-SFT	34	36	42	8.1	8.3	7.9
	AoT-SFT	66	64	71	24.8	26.0	33.1
	AoT-O3 (κ=0.0)	71	**68**	71	18.0	14.9	17.0
	AoT-O3	**73**	**68**	**75**	17.9	14.2	15.7
W-Ladder	CoT-SFT	11	5	14	6.3	5.5	6.6
	AoT-SFT	59	47	63	32.0	26.1	31.7
	AoT-O3 ($\kappa = 0.0$)	59	51	**66**	18.6	16.2	20.1
	AoT-O3	**61**	**52**	**66**	18.2	13.8	16.0

Table 2. Performance comparison of CoT-SFT, AoT-SFT, and AoT-O3 across different model sizes on Game of X, N-Puzzle and Word-Ladder benchmarks. Results show both accuracy (percentage of successfully solved problems) and solution length (average number of reasoning steps). AoT-O3 consistently achieves higher accuracy while requiring significantly fewer steps across all models and tasks.

words ranging from 4 to 10 letters in length. Test instances are generated by first selecting random word pairs of equal length from this corpus, then verifying that a valid solution path exists between them with lengths ranging from 4 to 10 steps. This formulation creates a challenging planning problem where the actions require language knowledge, and success requires careful navigation through the lexical graph while avoiding invalid words or circular paths.

5.2. Training and Testing Protocols

For both benchmarks, we implement a two-phase training approach. The supervised fine-tuning (SFT) phase consists of 500 training steps with a batch size of 128 and a learning rate of 1e-5. This is followed by the reinforcement learning phase using RLOO (REINFORCE leave-one-out), where we employ a smaller batch size of 32 due to VRAM constraints and a reduced learning rate of 1e-6. During the RL phase, we generate 4 samples per problem to estimate policy gradients while maintaining reasonable computational requirements. Additionally, we set $\beta = -0.5$ and $\kappa = 0.2$. Further details are given in Appendix B.

5.3. Results and Analysis

Table 2 presents comprehensive results comparing AoT-O3 against CoT-SFT and AoT-SFT baselines across three benchmarks and multiple model sizes. Our results demonstrate consistent improvements in both accuracy and solution efficiency. In the Game of X benchmark, AoT-O3 achieves substantial improvements across all models, with solution length reductions ranging from 51.3% to 62.4% while simultaneously improving accuracy by 11-17 percentage points. The most notable improvement is observed with Llama3-1B, reducing solution length from 31.9 to 9.7 steps while improving accuracy from 55% to 72%. For the N-Puzzle benchmark, we observe moderate but consistent improvements. AoT-O3 achieves solution length reductions between 27.8% and 46.0% while maintaining or improving accuracy. The smaller efficiency gains compared to Game of X can be attributed to N-Puzzle's larger state space and more complex state transitions. The Word-Ladder results are particularly interesting as they demonstrate AoT-O3's effectiveness in domains where actions must be derived from the LLM's prior knowledge of valid English words. Despite this additional complexity, AoT-O3 achieves a 47.1% reduction in solution length (from 26.1 to 13.8 steps) while improving accuracy from 47% to 52% on Llama3-1B. This suggests that our approach effectively combines the model's learned knowledge with efficient planning strategies. These improvements in efficiency do not come at the cost of accuracy - we observe consistent accuracy gains across all models and benchmarks. This indicates that our approach not only promotes more efficient solutions but also helps models develop more robust planning strategies through focused exploration of the solution space. The results are particularly promising for applications where context window limitations or computational resources are constrained, as AoT-O3 can achieve better performance with significantly fewer tokens.

6. Conclusion

In this work, we introduced AoT-O3, an RL-enhanced AoT framework that improves the efficiency of LLM-driven planning. By using a reward model that optimizes for both accuracy and conciseness, our approach reduces reasoning steps while maintaining quality. Results show up to 80% reduction in solution length across benchmarks, demonstrating how structured learning and reinforcement fine-tuning can make LLM planning more scalable and efficient.

Impact Statement

This work has significant environmental and accessibility implications for AI deployment. By reducing token usage by up to 80% while maintaining or improving performance, our method could substantially decrease the energy consumption and carbon footprint of large-scale LLM deployments. Given current LLM usage patterns, even a 30% reduction in tokens could save tens of gigawatt-hours annually, equivalent to the electricity usage of thousands of homes. However, such efficiency gains could also accelerate LLM adoption and lead to higher net resource consumption through the Jevons paradox. Additionally, while improved planning capabilities may enhance AI systems' problem-solving abilities, this could accelerate automation in ways that affect human employment. We recommend careful consideration of these tradeoffs as the technology is deployed, along with continued research into techniques that balance computational efficiency with societal impact.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Agency, I. E. Ai, cryptocurrency, and data centers: Projected energy consumption by 2026. *Data Center Frontier*, 2024. URL https://www.datacenterfrontier.com/energy/article/33038469/iea-study-sees-ai-cryptocurrency-doubling-data-center-energy-consumption-by-2026. Accessed: 2025-01-30.

Ahmadian, A., Cremer, C., Gallé, M., Fadaee, M., Kreutzer, J., Pietquin, O., Üstün, A., and Hooker, S. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Podstawski, M., Gianinazzi, L., Gajda, J., Lehmann, T., Niewiadomski, H., Nyczyk, P., et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17682–17690, 2024.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Earth.Org. Generative ai is exhausting the power grid. *Earth.Org*, 2024. URL https://earth.org/generative-ai-is-exhausting-the-power-grid/. Accessed: 2025-01-30.

Felner, A., Goldenberg, M., Sharon, G., Stern, R., Beja, T., Sturtevant, N., Schaeffer, J., and Holte, R. Partial-expansion a* with selective node generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pp. 471–477, 2012.

Huang, J. and Chang, K. C.-C. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*, 2022.

Jiang, J., Wang, F., Shen, J., Kim, S., and Kim, S. A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*, 2024.

Jin, M., Sel, B., Hardeep, F., and Yin, W. Democratizing energy management with llm-assisted optimization autoformalism. In *2024 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pp. 258–263. IEEE, 2024.

Kahneman, D. Thinking, fast and slow. *Farrar, Straus and Giroux*, 2011.

Kambhampati, S., Valmeekam, K., Guan, L., Verma, M., Stechly, K., Bhambri, S., Saldyt, L., and Murthy, A. Llms can't plan, but can help planning in llm-modulo frameworks. *arXiv preprint arXiv:2402.01817*, 2024a.

Kambhampati, S., Valmeekam, K., Guan, L., Verma, M., Stechly, K., Bhambri, S., Saldyt, L. P., and Murthy, A. B. Position: Llms can't plan, but can help planning in llm-modulo frameworks. In *Forty-first International Conference on Machine Learning*, 2024b.

Kamoi, R., Zhang, Y., Zhang, N., Han, J., and Zhang, R. When can llms actually correct their own mistakes? a

critical survey of self-correction of llms. *arXiv preprint arXiv:2406.01297*, 2024.

Katz, D. M., Bommarito, M. J., Gao, S., and Arredondo, P. Gpt-4 passes the bar exam. *Philosophical Transactions of the Royal Society A*, 382(2270):20230254, 2024.

Kim, G., Baldi, P., and McAleer, S. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*, 36, 2024.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.

Lei, B., Liao, C., Ding, C., et al. Boosting logical reasoning in large language models through a new framework: The graph of thought. *arXiv preprint arXiv:2308.08614*, 2023.

Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Eccles, T., Keeling, J., Gimeno, F., Dal Lago, A., et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

Long, J. Large language model guided tree-of-thought. *arXiv preprint arXiv:2305.08291*, 2023.

Ma, X., Mishra, S., Beirami, A., Beutel, A., and Chen, J. Let's do a thought experiment: Using counterfactuals to improve moral reasoning. *arXiv preprint arXiv:2306.14308*, 2023.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2024.

Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., and Zettlemoyer, L. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022.

Mnih, V. Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*, 2016.

Nye, M., Andreassen, A. J., Gur-Ari, G., Michalewski, H., Austin, J., Bieber, D., Dohan, D., Lewkowycz, A., Bosma, M., Luan, D., et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Paul, D., Ismayilzada, M., Peyrard, M., Borges, B., Bosselut, A., West, R., and Faltings, B. Refiner: Reasoning feedback on intermediate representations. *arXiv preprint arXiv:2304.01904*, 2023.

Radford, A. Improving language understanding by generative pre-training. 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Rasley, J., Rajbhandari, S., Ruwase, O., and He, Y. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 3505–3506, 2020.

Russell, S. J. and Norvig, P. Artificial intelligence: A modern approach. 1995.

Samsi, S., Zhao, D., McDonald, J., Li, B., Michaleas, A., Jones, M., Bergeron, W., Kepner, J., Tiwari, D., and Gadepally, V. From words to watts: Benchmarking the energy costs of large language model inference. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–9. IEEE, 2023.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In Bach, F. and Blei, D. (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1889–1897, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schulman15.html.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sel, B., Shanmugasundaram, P., Kachuee, M., Zhou, K., Jia, R., and Jin, M. Skin-in-the-game: Decision making via multi-stakeholder alignment in llms. *arXiv preprint arXiv:2405.12933*, 2024a.

Sel, B., Tawaha, A., Khattar, V., Jia, R., and Jin, M. Algorithm of thoughts: Enhancing exploration of ideas in large language models. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 44136–44189. PMLR, 21–27 Jul 2024b. URL https://proceedings.mlr.press/v235/sel24a.html.

Sel, B., Jia, R., and Jin, M. Llms can plan only if we tell them. 2025. URL https://arxiv.org/abs/2501.13545.

Valmeekam, K., Marquez, M., and Kambhampati, S. Can large language models really improve by self-critiquing their own plans? *arXiv preprint arXiv:2310.08118*, 2023.

Valmeekam, K., Stechly, K., and Kambhampati, S. Llms still can't plan; can lrms? a preliminary evaluation of openai's o1 on planbench. *arXiv preprint arXiv:2409.13373*, 2024.

Vaswani, A. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Webb, P. *System 3 Thinking: How to Choose Wisely when Facing Doubt, Dilemma, Or Disruption*. Intentional Training Concepts Pty Limited, 2021. ISBN 9781922553560. URL https://books.google.com/books?id=M1G6zgEACAAJ.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Williams, R. J. and Peng, J. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Yao, Y., Li, Z., and Zhao, H. Beyond chain-of-thought, effective graph-of-thought reasoning in language models. *arXiv preprint arXiv:2305.16582*, 2023.

Yoshizumi, T., Miura, T., and Ishida, T. A* with partial expansion for large branching factor problems. In *AAAI/IAAI*, pp. 923–929, 2000.

Yuan, A., Coenen, A., Reif, E., and Ippolito, D. Wordcraft: story writing with large language models. In *Proceedings of the 27th International Conference on Intelligent User Interfaces*, pp. 841–852, 2022.

Zelikman, E., Lorch, E., Mackey, L., and Kalai, A. T. Self-taught optimizer (stop): Recursively self-improving code generation. *arXiv preprint arXiv:2310.02304*, 2023.

Zelikman, E., Harik, G., Shao, Y., Jayasiri, V., Haber, N., and Goodman, N. D. Quiet-star: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*, 2024.

Zhang, Z., Zhang, A., Li, M., and Smola, A. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*, 2022.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q., et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., Dey, A. K., et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A. Energy and Token Usage Calculations

In this appendix, we provide a detailed estimation of the current token usage in ChatGPT and the potential energy savings from reducing token counts. All values and references herein are approximate but illustrate the magnitude of efficiency gains possible at scale.

A.1. Estimating Token Usage

User Base and Queries. As of December 2024, ChatGPT has over 300 million weekly active users, with users sending more than 1 billion messages daily.[2] Assuming each message involves an average of 400 tokens — encompassing both user input and model output — we can estimate the total token usage.

Total Daily and Yearly Token Count. Given over 1 billion messages daily, the total daily token amounts to 400 billion. Extrapolating to a full year results in approximately 146 trillion tokens per year.

A.2. Energy Usage per Token

Energy consumption per token varies with model size, hardware, and data-center efficiency. Research estimates that models like LLaMA 65B consume about 3–4 joules per token (Samsi et al., 2023). Since 1 joule $= 2.77778 \times 10^{-7}$ kWh, 3–4 joules per token converts to roughly 8.33×10^{-7} kWh/token to 1.11×10^{-6} kWh/token. We use the midpoint 9.72×10^{-7} kWh/token for our calculations.

A.3. Daily and Annual Energy Consumption

Using the midpoint estimate, we see 400 billion tokens/day $\times\ 9.72 \times 10^{-7}$ kWh/token ≈ 388.8 MWh/day.

Over one year, we see 388.8 MWh/day $\times\ 365 \approx 141.9$ GWh/year.

A.4. Potential Savings from Token Reduction

Reducing token usage by 30% results in approximately 42.6 GWh/year.

If we cut token counts by half, we see approximately 71 GWh/year savings in energy usage.

A.5. Contextualizing the Energy Savings

Residential Electricity Usage. According to the U.S. Energy Information Administration (EIA), the average annual electricity consumption for a U.S. residential utility customer was about 10,791 kWh in 2022.[3] Hence:

- **42.6 GWh/year** savings $\approx$ electricity usage of $\frac{42.6 \times 10^6 \text{ kWh}}{10{,}791 \text{ kWh/home}} \approx 3{,}948$ homes for a year.
- **71 GWh/year** savings $\approx$ electricity usage of about $\frac{71 \times 10^6 \text{ kWh}}{10{,}791 \text{ kWh/home}} \approx 6{,}580$ homes for a year.

Carbon Emission Reductions. Using an estimate of 0.81 pounds (0.367 kg) of CO_2 emitted per kWh in the U.S.[4]:

- 42.6 GWh $\rightarrow$ 42.6 $\times 10^6$ kWh $\times$ 0.367 kg CO_2/kWh $\approx$ 15,634 metric tons CO_2.
- 71 GWh $\rightarrow$ 71 $\times 10^6$ kWh $\times$ 0.367 kg CO_2/kWh $\approx$ 26,057 metric tons CO_2.

These quantities correspond to removing thousands of gasoline-powered cars from the road or preserving thousands of acres of forest in terms of carbon offsets (see Fig. 3).[5]

[2] https://www.theverge.com/2024/12/4/24313097/chatgpt-300-million-weekly-users
[3] https://www.eia.gov/tools/faqs/faq.php?id=97&t=3
[4] https://www.eia.gov/tools/faqs/faq.php?id=74&t=11
[5] https://www.epa.gov/energy/greenhouse-gas-equivalencies-calculator

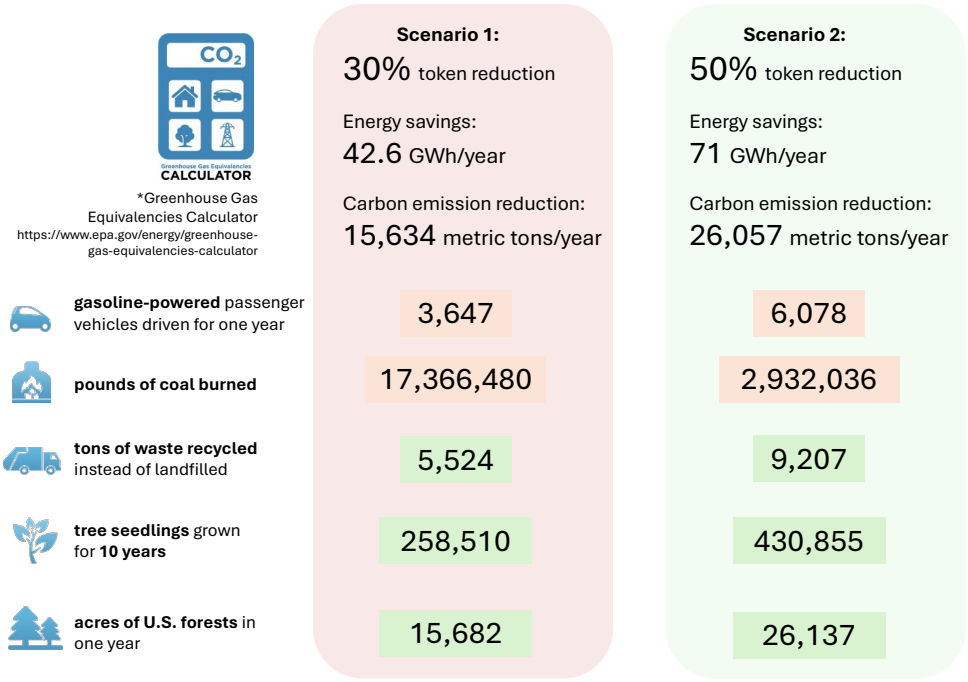

Figure 3. Illustration of two scenarios—a 30% (pink) vs. 50% (green) reduction in token usage—and illustrates their equivalent environmental impacts (e.g., cars off the road, coals burned, waste recycled, seedlings grown, acres of forest). These numbers contextualize the calculations by showing how modest efficiency gains can yield significant benefits for emissions and resource consumption.

Overall, these estimates highlight the potential environmental and cost impact that arises from large-scale LLM inference. Even moderate gains in token efficiency (e.g., 30–50%) can translate to tens of gigawatt-hours saved annually and correspondingly meaningful CO_2 reductions.

Key Assumptions and Limitations In these estimates, data center energy efficiency (often quantified by Power Usage Effectiveness, or PUE) plays a major role. An ultra-efficient center with a PUE near 1.1 uses roughly 10% extra overhead (e.g., cooling and networking), whereas a PUE above 1.5 can raise total consumption by 50% or more. Thus, even if token usage shrinks by 30–50%, the real reduction in total energy may be proportionally less depending on the facility's PUE. Moreover, the uniform-token-savings assumption—that energy reduction scales linearly with fewer tokens—overlooks fixed overheads (e.g., loading the model into memory each query) that may blunt the full benefits of shorter outputs. Finally, unaccounted training energy can weigh in daily inference usage when models are re-trained frequently or undergo large-scale fine-tuning cycles. Hence, while reducing tokens helps, it alone cannot capture AI's entire environmental footprint. We also did not account for the complex Jevons paradox, i.e., technological improvements increase the efficiency of resource use, which may paradoxically lead to an overall increase in the consumption of that resource.

B. Experiment Details

This appendix provides comprehensive details about our experimental setup, hyperparameters, and implementation choices to ensure reproducibility of our results.

B.1. Model Specifications

We conducted experiments using three open-source language models:

- Gemma2-2B: 2 billion parameters, using bfloat16 data type.
- Llama3-1B: 1 billion parameters, using bfloat16 data type.
- Llama3-3B: 3 billion parameters, using bfloat16 data type.

All models were trained on 8x NVIDIA H100 GPUs with 80GB memory. For multi-GPU training, we utilized DeepSpeed (Rasley et al., 2020) with a suitable gradient accumulation step of to maintain effective batch sizes while managing memory constraints for the various models we trained.

B.2. Dataset Construction

For each benchmark, we constructed datasets following these specifications:

Game of X

- Training set: 64,000 examples
- Test set: 100 examples
- Numbers sampled uniformly from [1, 100]
- Target values sampled uniformly from [1, 100]
- Filtered to ensure each problem has at least one valid solution

N-Puzzle

- Training set: 64,000 examples
- Test set: 100 examples
- Initial states generated with 80-120 random moves from goal state
- Manhattan distance from initial to goal state ranging from 15 to 30

Word Ladder

- Training set: 32,000 examples
- Test set: 100 examples
- Word length range: 4-10 characters
- Solution path length range: 4-10 steps
- Dictionary: NLTK Words Corpus (filtered for common English words)

B.3. Training Protocol

Our training process consisted of two phases: Supervised Fine-tuning (SFT) and Reinforcement Learning (RL). Here we detail the specific parameters and protocols for each phase.

B.3.1. Supervised Fine-tuning Phase

Optimization Parameters

- Optimizer: AdamW
- Base learning rate: 1e-5
- Weight decay: 0.01
- Gradient clipping: 0.1 (max norm)
- Batch size: 128 (effective, after gradient accumulation)

- Training steps: 500
- Warm-up steps: 50
- Learning rate scheduler: Cosine annealing

Implementation Details

- Mixed precision training (BF16)
- Gradient checkpointing enabled
- Token sequence length: 2048
- Padding: Dynamic batching with attention masking

B.3.2. Reinforcement Learning Phase

We utilized trl library from huggingface, however, we needed to have a custom RLOO trainer implementation with minor but crucial differences to the original library due to it being designed for LLM reward models.

RLOO Implementation

- Algorithm: REINFORCE leave-one-out
- Base learning rate: 1e-6
- Batch size: 32
- Samples per problem: 4
- KL penalty coefficient: 0.1
- Value loss coefficient: 0.5

Reward Model Parameters

- Step penalty factor (α): 0.02
- Minimum reward cutoff (β): -0.5
- Solution path bonus (κ): 0.2

Training Schedule

- Total steps: 50
- Warm-up steps: 50
- Learning rate schedule: Cosine annealing

B.4. Evaluation Protocol

During evaluation, we used the following settings:

Inference Parameters

- Temperature: 0.0
- Top-p (nucleus sampling): 0.0
- Maximum new tokens: 1024
- Repetition penalty: No