Is Repeatedly Solving the Same Problem Necessary? Rethinking Task Reasoning through Action Prototype Learning

Anonymous ACL submission

Abstract

Large language models (LLMs) demonstrate remarkable proficiency across a wide array of tasks but often struggle with specialized problem-solving and practical reasoning, especially in domains such as mathematics. Existing approaches frequently rely on solving the same problem multiple times (e.g., 20 iterations) to achieve high precision. We argue that such redundancy is unnecessary, as humans rarely solve problems this way. To address this, we propose action prototype learning for task reasoning, where strategies are systematically organized into discrete action prototypes, each associated with a semantic key and prior knowledge, enabling efficient task alignment and reasoning. Additionally, we design a contradiction-based answer evaluation mechanism to identify logical inconsistencies with problem data, enhancing solution precision. We also develop an action-matching inference mechanism that retrieves relevant prior knowledge, significantly reducing token consumption while improving inference performance. By leveraging efficient reasoning strategies, our method requires only a single pass to achieve high-quality results, minimizing excessive computations. Extensive evaluations across two datasets show that our approach reduces token usage by approximately 68.5% compared to self-consistency (SC) methods while maintaining robust reasoning capabilities. This highlights the effectiveness of leveraging prior knowledge to refine LLM reasoning, making it both efficient and practical.

1 Introduction

With the rapid advancement of large language models (LLMs), techniques such as Chain of Thought (CoT) and self-consistency (SC) have been proposed to enhance their reasoning capabilities. CoT enables step-by-step reasoning, allowing models to articulate intermediate steps. However, the incremental generation process introduces inherent randomness, especially when sampling strategies like temperature or Top-p sampling are used. This randomness often results in inconsistent outputs, limiting the reliability of CoT for tasks that require precise reasoning, such as mathematical problem-solving.

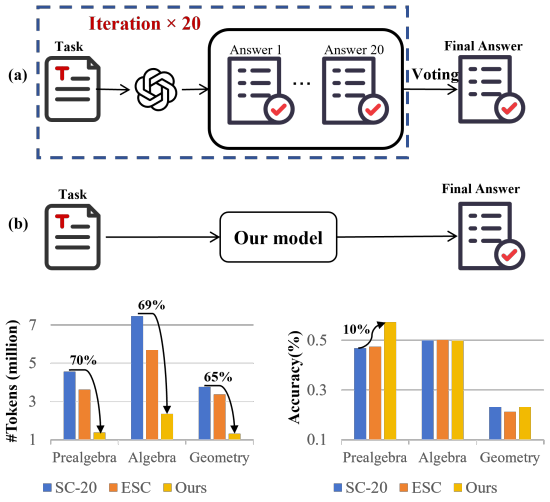

Figure 1: SC Models vs. Our Model for Task Reasoning. (a) Pipeline of SC Models: In traditional approaches, multiple passes through the model (e.g., 20 iterations) are typically used to refine the reasoning process. (b) The Proposed Method: In contrast, our proposed method only requires a single pass for task reasoning. Specifically, for Prealgebra problems, our model not only improves accuracy by around 10%, but also achieves a substantial reduction in token consumption—up to 70% by leveraging a more efficient and effective structure.

To address these inconsistencies, SC generates multiple answers in parallel and selects the most frequent one as the final output, which improves reliability. However, this method significantly increases token consumption, as even generating a single answer with CoT requires multiple reasoning steps. SC amplifies this cost by repeatedly generating answers for the same question. Early-stopping self-consistency (ESC) mitigates this by detecting convergence during repeated answers, reducing token usage. However, because its underlying logic remains probabilistic, ESC struggles to further re-

duce token consumption.

To this end, inspired by human reasoning, we propose incorporating prior knowledge into LLMs to enhance both efficiency and accuracy. Just as humans refine their reasoning by leveraging insights from similar problems or lessons learned from past mistakes (Wu et al., 2024), we design a proof-by-contradiction approach, where candidate answers are tested by substituting them back into the problem to check for logical inconsistencies. This method outputs a Boolean value (True or False) to verify reasoning accuracy. To further enhance efficiency, we design an action knowledge space that distills problem-solving techniques into distinct action prototypes, each with semantic keys and associated prior knowledge. During inference, the model selects the most relevant action prototypes based on similarity scores, enabling task-specific reasoning while minimizing redundancy. We experimentally compare our method with SC methods in Fig. 1, where it is evident that our approach only requires a single pass for task reasoning, showcasing a significant improvement in token efficiency and accuracy increase.

In summary, our contributions are three-fold:

(1) Compared to solving the same problem repeatedly, we argue that such redundancy is inefficient, as humans rarely solve a problem multiple times in this way. To address this issue, we propose the concept of action knowledge space and develop action prototype learning to handle task reasoning for LLMs. It distills and explicitly represents problem-solving techniques as action prototypes, each connected to a semantic key and relevant prior knowledge, enhancing both the efficiency and effectiveness of LLM inference.

(2) We present a contradiction-based answer evaluation mechanism that verifies candidate answers by substituting them back into the original problem and checking for logical conflicts. This approach enhances the model's reasoning accuracy and robustness, particularly in complex tasks.

(3) We propose an efficient reasoning mechanism that aligns tasks with relevant actions from the action knowledge space. By integrating the prior knowledge of the matched actions with the contradiction-based evaluation mechanism, the model refines its answers. This greatly reduces token consumption while improving both inference efficiency and reasoning performance.

2 Related Work

2.1 Chain of Thought (CoT) Reasoning

Chain of Thought (CoT) (Wei et al., 2022) reasoning enhances large language models (LLMs) by prompting intermediate reasoning steps, enabling better performance on tasks such as mathematical problem-solving (Patel et al., 2021; Miao et al., 2020), logical reasoning (Geva et al., 2021; Yang et al., 2018), and commonsense inference (Talmor et al., 2019; Aggarwal et al., 2021). Extensions of CoT include automating reasoning step generation and integrating external tools to further improve performance. However, CoT's reliance on incremental generation introduces randomness through sampling strategies like temperature, Top-k, or Top-p sampling (Touvron et al., 2023; Achiam et al., 2023), which can lead to inconsistent reasoning paths and incorrect answers. These limitations reduce CoT's reliability for tasks requiring high accuracy and consistency.

2.2 Self-Consistency-Based Reasoning

Self-Consistency (SC) (Wang et al., 2022) improves CoT by generating multiple reasoning paths in parallel and selecting the most frequent answer (Yang et al., 2024), leveraging diversity to enhance reliability. However, SC significantly increases token consumption, as each reasoning path is generated step-by-step. Early-Stopping Self-Consistency (ESC) (Li et al., 2024) addresses this by detecting consecutive identical answers and terminating generation early, reducing computational cost while maintaining accuracy. Despite these improvements, both SC and ESC rely on repeated answering, which can lead to redundancy and diminishing returns, particularly when answers from multiple attempts show little variation. This highlights the need for more efficient inference strategies that balance accuracy, diversity, and computational cost.

2.3 Contradiction-Based Answer Evaluation

Contradiction-based evaluation substitutes candidate answers back into the problem to check for conflicts with the problem's information, inspired by proof-by-contradiction principles (Lewkowycz et al., 2022; Xie et al., 2024; Chen et al., 2024; Yao et al., 2024). This approach has been applied to tasks like verifying logical consistency or validating mathematical solutions. Methods such as Self-refine (Madaan et al., 2024) and Self-check (Miao et al., 2023) enable models to evaluate and

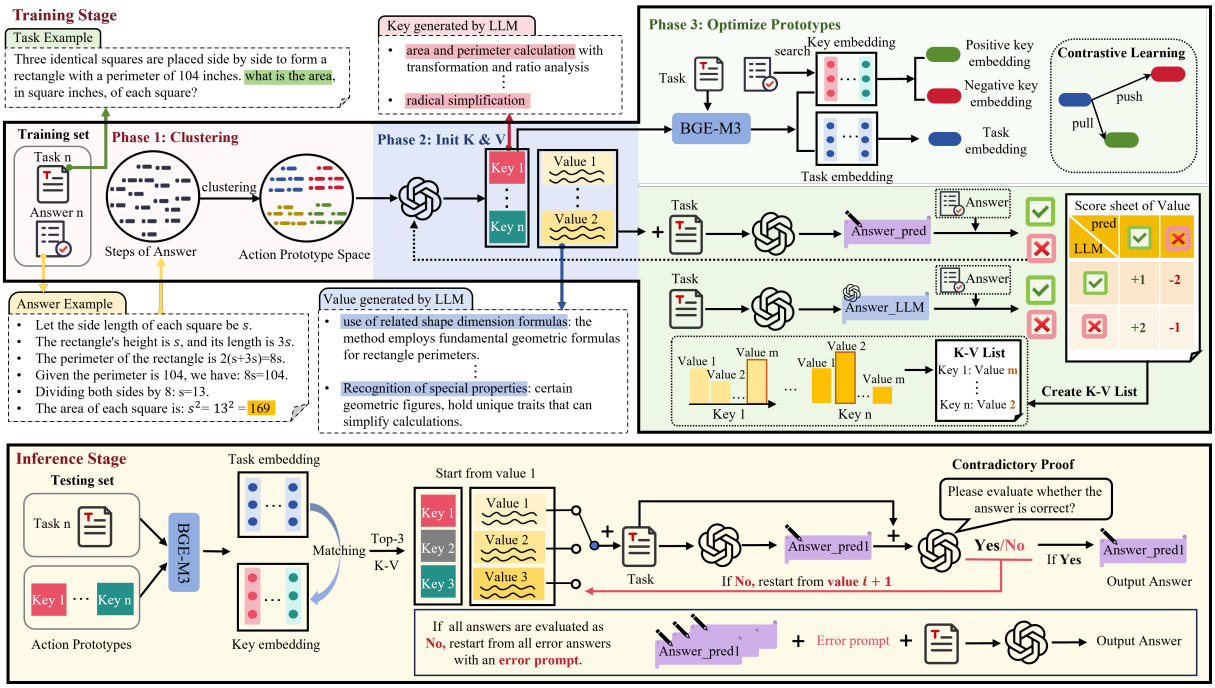

Figure 2: The architecture of the proposed method. It consists of two stages: training and inference. In the training stage, problem-solving techniques are distilled and explicitly represented as action prototypes, each associated with a semantic key and relevant prior knowledge. During the inference stage, we design an efficient contradiction-based reasoning method, which aligns tasks with the most relevant actions from the action knowledge space.

adjust their outputs without relying on external information (Shinn et al., 2024; Yeo et al., 2024). However, these methods often struggle with insufficient context for effective corrections. Our approach integrates contradiction-based evaluation with action prototypes (Jiang et al., 2022), allowing for dynamic error correction and improved reasoning accuracy by aligning tasks with relevant prior knowledge.

2.4 Task-Specific Knowledge and Action Matching

Task-specific knowledge is a well-established strategy for improving reasoning performance. Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Gao et al., 2023) retrieves relevant external knowledge to assist in reasoning, while problem decomposition techniques break down complex tasks into simpler subproblems (Khot et al., 2022; Zhang et al., 2021; Perez et al., 2020; Radhakrishnan et al., 2023; Dua et al., 2022). However, these approaches often rely on external resources or annotations, which may not always be available. Recent advances in contrastive learning have shown promise in associating tasks with relevant knowledge representations, but few studies explicitly model problem-solving techniques as reusable actions. This lack of explicit representation limits LLMs' ability to generalize across tasks and efficiently apply prior knowledge. Addressing these challenges requires integrating reusable actions, contradiction-based evaluation, and efficient token usage to balance accuracy, efficiency, and consistency (Yu et al., 2024; Yao et al., 2024).

3 Method

To address the limitations of iterative calculations in existing inference strategies for large language models (LLMs), we propose a novel method that integrates an action knowledge space, a random walk-based action prototype optimization algorithm, and a contradiction-based reasoning mechanism. This framework systematically models reusable problem-solving strategies as structured actions, optimizes their effectiveness through training, and incorporates them into the reasoning process, which enhances both accuracy and efficiency.

3.1 The Architecture

The architecture of the proposed method is presented in Fig. 2. It consists of two stages: training and inference. During the training stage, reasoning steps are clustered into action prototypes, represented by action labels (K) and action values (V). A

semantic embedding model, bge-m3, is fine-tuned with contrastive learning to capture the functional similarities between tasks and action prototypes, enabling accurate task-action matching. Additionally, a random walk-based algorithm is employed to optimize the action prototypes, improving their quality and ensuring they are closely aligned with specific tasks. In the inference phase, tasks are matched to the most relevant action prototypes. A contradiction-based reasoning mechanism then evaluates each generated answer for logical consistency. If contradictions persist after three iterations, the conflicting answers are treated as negative examples, which helps refine the final output. The framework utilizes LLaMA3-8B (Meta, 2024) as the backbone LLM, effectively balancing accuracy and efficiency throughout the reasoning process.

3.2 Action Knowledge Space

The action knowledge space is a key component designed to distill and represent reusable problem-solving techniques derived from the training data (Shridhar et al., 2022). It consists of two main elements: action labels, which are semantic identifiers that categorize the various actions involved in problem-solving, and action prototypes, which refers to prior knowledge distilled from reasoning steps. Together, these form a structured repository that aids LLMs in solving new tasks by providing them with efficient, contextually relevant strategies. This space ensures that LLMs can quickly access and apply effective actions without the need for redundant recalculations, significantly improving reasoning efficiency.

For each task in the training set, we leverage its corresponding answer, typically provided in a Chain of Thought (CoT) format. By segmenting the CoT answer into individual sentences, we isolate the distinct reasoning steps that lead to the solution of the task. These reasoning steps are then aggregated across all tasks in the training set, forming a comprehensive collection of problem-solving strategies. This compilation of reasoning steps serves as the foundation for building the action knowledge space, where each step can be categorized and linked to relevant task-specific action prototypes, enabling more efficient and effective task reasoning.

To identify common patterns and problem-solving techniques, we apply unsupervised clustering to the extracted reasoning steps. Each resulting cluster represents a distinct action, corresponding to a specific problem-solving skill or method. For every cluster, we assign an action label, which serves as a semantic identifier, and an action prototype, which encapsulates the prior knowledge distilled from the reasoning steps within that cluster. While the action labels remain fixed after initialization, the action prototype undergoes further optimization during the training process. This ensures that the action knowledge is continuously refined, enhancing its utility in supporting the LLM's reasoning and improving the overall task-solving efficiency.

3.3 Task-Action Matching via Contrastive Learning

To effectively match tasks with the most relevant action prototypes in the action knowledge space, we fine-tune a pre-trained sentence embedding model (e.g., bge-m3) using contrastive learning. Unlike traditional semantic similarity models that focus on surface-level matching, our fine-tuning process is specifically designed to capture the functional relationship between tasks and actions. This allows the model to identify actions that are functionally relevant to solving a given task, even if their semantic representations differ significantly. By emphasizing functional relevance rather than purely semantic similarity, the model is better equipped to match tasks with the most appropriate problem-solving techniques, improving task reasoning accuracy and efficiency.

Contrastive Learning Objective. The fine-tuning process follows a contrastive learning framework (Khosla et al., 2020; Tian et al., 2020; Le-Khac et al., 2020; Chen et al., 2020), in which the model is trained to map both tasks and action labels into a shared embedding space. In this space, tasks are aligned with actions that represent effective problem-solving techniques, rather than simply semantically similar concepts. To achieve this alignment, we construct training triplets of the form (t, a^+, a^-), where t is the embedding of a task, a^+ is the embedding of a positive action label that corresponds to an action prototype offering a relevant problem-solving technique for the task, and a^- is the embedding of a negative action label, representing an action prototype that is not functionally relevant to the task.

Subsequently, the contrastive loss function can be defined as in Eq. (1):

$$\mathcal{L}_{\text{cont}} = -\log \frac{\exp\left(\frac{\text{sim}(t,a^+)}{\tau}\right)}{\exp\left(\frac{\text{sim}(t,a^+)}{\tau}\right)+\sum_{a^-}\exp\left(\frac{\text{sim}(t,a^-)}{\tau}\right)} \tag{1}$$

where $\text{sim}(\cdot,\cdot)$ represents the cosine similarity between two embeddings, and τ is a temperature parameter that controls the sharpness of the similarity distribution. This loss function encourages the model to maximize the similarity between the task embedding t and the positive action label embedding a^+, while minimizing the similarity between t and the negative action label embedding a^-, enabling the model to effectively associate tasks with relevant actions.

Action Prototype Learning. To construct the training data, we utilize the action knowledge space and the training task set. For each task, we extract the corresponding answer, typically provided in a Chain of Thought (CoT) format, and break it down into individual reasoning steps. These reasoning steps have been pre-assigned to specific action prototypes through the unsupervised clustering process (Likas et al., 2003), ensuring that each step is naturally associated with a corresponding prototype in the action knowledge space.

For each reasoning step in a task, the corresponding action prototype is treated as a positive example (a^+), as it represents the relevant problem-solving technique for that step. On the other hand, action prototypes that do not include any reasoning steps from the task are treated as negative examples (a^-). By constructing training triplets (t, a^+, a^-) in this way, the model learns the functional relationship between tasks and actions. This method utilizes the inherent structure of the action knowledge space, eliminating the requirement for additional matching or alignment procedures. It ensures that the model can directly associate tasks with their most relevant problem-solving techniques based on their functional relevance, streamlining the learning process.

Fine-Tuning Phase. The sentence embedding model (e.g., bge-m3) is initialized with pre-trained weights to take advantage of its general-purpose semantic representation capabilities. To capture the functional relationship between tasks and actions, the model is fine-tuned using the constructed triplets (t, a^+, a^-) along with the contrastive loss function. During fine-tuning (Hu et al., 2021), the model adjusts the embedding space so that the task embeddings are positioned closer to the embeddings of their functionally relevant actions while being pushed further apart from irrelevant actions. This optimization process ensures that the model learns to align tasks with actions based on their functional relevance, moving beyond surface-level semantic similarity.

Inference Phase. In the inference phase, the fine-tuned model is employed to identify the most relevant action prototypes for a given task. The task is first encoded into an embedding vector using the fine-tuned model. Meanwhile, the embeddings of all action labels in the action knowledge space are precomputed and stored. The cosine similarity between the task embedding and each action label embedding is then computed. Based on these similarity scores, the top three action labels with the highest similarity are selected as the most relevant action prototypes for the task. These selected action prototypes represent the prior knowledge that best aligns with the task's requirements and are used to guide the reasoning process.

3.4 Random Walk-Based Action Prototype Optimization

While the initial action prototype in the action knowledge space provides a useful starting point, its utility in assisting LLM reasoning may vary across different tasks. To enhance the effectiveness of this action knowledge, we propose a random walk-based optimization algorithm (Wang et al., 2024). This algorithm dynamically adjusts the quality and relevance of each action prototype by exploring and refining the action space iteratively. Through this process, the action prototypes are refined to become more task-specific, ensuring that only the most relevant and high-quality prototypes are used during inference.

For each task in the training set, we first identify the most relevant action prototypes by matching the task with the action labels in the knowledge space. Once selected, we generate multiple candidate descriptions representing different aspects or formulations of the action knowledge. These descriptions augment the task, providing additional context and guidance for the LLM during reasoning. To evaluate their utility, we compare the baseline answer (produced without action knowledge) and the knowledge-augmented answers (generated with the descriptions) against the ground truth.

The evaluation process adjusts the scores of

Algorithm 1 Random Walk-Based Action Prototype Optimization Algorithm

Input: Action space $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ with initial prototypes $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$; Training set $\mathcal{D}_t$ with M tasks; Ground truth answers $\mathcal{G}$ for tasks in $\mathcal{D}_t$; Maximum attempts per task τ_{max}.
Output: Optimized Action Prototypes $\mathcal{P}^* = \{P_1^*, P_2^*, \ldots, P_n^*\}$.

for each task $T_i \in \mathcal{D}_t$ **do**
  Find relevant action A_i for T_i in $\mathcal{A}$
  if $\mathcal{C}_{A_i}$ is not initialized **then**
    Initialize candidate set $\mathcal{C}_{A_i} \leftarrow \{P_j \in \mathcal{P}\}$
  end if
  Initialize attempt counter $\tau \leftarrow 0$
  Initialize a counter for visited prototypes $v \leftarrow 0$
  while $\tau < \tau_{\text{max}}$ **and** $v < |\mathcal{C}_{A_i}|$ **do**
    Select the highest-scoring prototype P_i from $\mathcal{C}_{A_i}$, excluding previously visited prototypes
    Knowledge-Augmented Answer $\leftarrow \text{LLM}(P_i, T_i)$
    if *Knowledge-Augmented Answer* matches $\mathcal{G}[T_i]$ **then**
      Increment score for P_i
      break
    else
      Decrement score for P_i
      Generate a new prototype $P_i' \leftarrow$ Refine P_i
      Add P_i' to candidate set $\mathcal{C}_{A_i}$
    end if
    Increment visited counter $v \leftarrow v + 1$
    Increment attempt counter $\tau \leftarrow \tau + 1$
  end while
end for
Update $\mathcal{P}$ by selecting the highest-scoring prototype from each candidate set $\mathcal{C}_{A_i}$
return Optimized Action Prototypes $\mathcal{P}^*$

candidate descriptions based on their relative performance. Candidate descriptions that enhance the accuracy of the LLM's answers are rewarded, while those that degrade performance are penalized. The magnitude of these adjustments depends on whether the baseline answer is correct or incorrect, with greater emphasis placed on cases where the knowledge-augmented answer corrects an initially incorrect baseline answer. This iterative scoring process is applied to all tasks in the training set, enabling the algorithm to continuously refine the action knowledge. As the process progresses, the action knowledge for each prototype is fine-tuned, optimizing its effectiveness in guiding LLM reasoning. At the end of the optimization cycle, the action knowledge is more precise and better aligned with the task requirements. The detailed steps of this optimization process are outlined in Algorithm 1.

3.5 Contradiction-Based Reasoning

To ensure logical consistency and accuracy in the reasoning process, we integrate a contradiction-based mechanism inspired by the principle of proof by contradiction. This method leverages the capabilities of LLMs to assess the logical coherence of their outputs. When inconsistencies or contradictions are identified in the initial response, the mechanism iteratively refines the answer by detecting and resolving conflicting elements. This process allows the model to gradually enhance its reasoning, ensuring the final output remains logically consistent and more likely to be accurate. By utilizing contradiction detection, we strengthen the robustness of the reasoning process, enabling the model to self-correct and refine its answers through logical validation.

Proof by contradiction is a well-established technique in formal reasoning, where an assumption is tested by deriving its logical consequences and identifying any contradictions with established facts. In our framework, the task and the action knowledge corresponding to the most relevant action prototypes (determined through task-action matching) are provided as input to the LLM to generate an initial answer. This answer is then evaluated by substituting it into the problem context to identify any potential contradictions. If no contradictions are found, the answer is accepted as the final result. However, if a contradiction is detected, the reasoning process is iteratively refined by incorporating additional action knowledge or adjusting the reasoning steps, ensuring that the final output is logically consistent and accurate.

This iterative mechanism not only ensures that the final answer adheres to the logical constraints of the problem but also strengthens the robustness of the reasoning process by systematically eliminating invalid solutions. By integrating proof-by-contradiction principles into LLM reasoning, our framework offers a scalable and efficient solution for complex tasks—such as mathematical problem-solving and logical inference—where consistency and accuracy are paramount.

4 Experiments

4.1 Datasets

We evaluate our method on two datasets: GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021). GSM8K is a widely recognized benchmark for arithmetic reasoning tasks, designed to evaluate the ability to perform multi-step numerical reasoning. MATH, on the other hand, is a comprehensive dataset that encompasses a wide variety of mathematical domains. It is further divided into subcategories, including algebra, geometry, num-

	GSM8k		MATH-probability		MATH-geometry		MATH-number theory		MATH-precalculus	
	Accuracy	Tokens	Accuracy	Tokens	Accuracy	Tokens	Accuracy	Tokens	Accuracy	Tokens
COT	0.774	**0.30m**	0.166	**0.18m**	0.169	**0.19m**	0.167	**0.21m**	0.172	**0.33m**
SC-10	0.835	3.03m	0.224	1.83m	0.200	1.88m	0.267	2.05m	0.243	3.16m
SC-20	**0.840**	6.06m	**0.283**	3.64m	**0.232**	3.76m	**0.291**	4.14m	**0.262**	6.26m
ESC	**0.838**	2.19m	0.274	3.31m	**0.213**	3.37m	**0.300**	3.69m	**0.254**	5.59m
Ours	**0.838**	**1.43m**	**0.258**	**1.36m**	**0.232**	**1.32m**	0.270	**1.22m**	0.196	**2.20m**

Table 1: Performance evaluation of the proposed method against state-of-the-art methods on GSM8k, MATH-probability, MATH-geometry, MATH-number theory, and MATH-precalculus. The best performance is highlighted in bold red, and the second-best performance is highlighted in bold blue.

ber theory, precalculus, probability, prealgebra, and intermediate algebra. These subcategories enable a fine-grained evaluation of reasoning capabilities across distinct mathematical topics.

4.2 Baselines

To assess the performance of our method, we compare it against three baseline approaches. The first baseline is Chain-of-Thought (COT) prompting (Wei et al., 2022), which generates reasoning steps sequentially without incorporating additional sampling or optimization techniques. The second baseline is Self-Consistency (SC) prompting (Wang et al., 2022), which samples multiple reasoning paths and determines the final answer via majority voting. Two variants of SC are evaluated, denoted as SC-10 and SC-20, corresponding to sampling 10 and 20 reasoning paths, respectively. The third baseline is Early-Stopping Self-Consistency (ESC) (Li et al., 2024), which improves upon SC by introducing a sliding-window mechanism to detect consecutive identical answers. Once a confident majority is detected, the generation process is terminated early to reduce computational cost.

4.3 Evaluation Metrics

The methods are evaluated using two key metrics. The first metric is accuracy, which represents the proportion of correctly solved problems within each dataset or subcategory. The second metric is token consumption, which measures the total number of tokens generated during the reasoning process and reflects the computational cost associated with each method.

4.4 Results on GSM8K and MATH

We conduct experiments on GSM8K and MATH, and the experimental results are summarized in Table 1 and Table 2.

On the GSM8K dataset, our method achieves competitive accuracy while significantly reducing token consumption compared to the baselines. Specifically, our method attains an accuracy of 0.838, which is higher than SC-10 (0.824) and comparable to SC-20 (0.841). In terms of token consumption, our method reduces the total token to 1.43 million tokens, which is a 52.8% reduction compared to SC-10's 3.03 million tokens and a 70.3% reduction compared to SC-20's 4.81 million tokens. On the MATH dataset, our method consistently achieves competitive accuracy while maintaining a low computational cost. On average, our model reduces the token count by 67.1% compared to SC-20 and 61.5% compared to ESC, with an accuracy drop of less than 1%. This substantial reduction in token consumption highlights the efficiency of our approach, particularly in tasks where both accuracy and computational cost are critical considerations.

In the subcategory evaluation, for instance, in the prealgebra subcategory, our method achieves an accuracy of 0.612, which is significantly higher than SC-20 (0.550) and ESC (0.556). Our token consumption is reduced to 2.12 million tokens, compared to 6.23 million tokens for SC-20 (a reduction of 65.9%) and 4.51 million tokens for ESC (a reduction of 53.0%). In other subcategories, such as probability, our method achieves an accuracy of 0.258 with a token consumption of 1.36 million tokens, which is lower than SC-10 (1.83 million tokens, a reduction of 25.7%), SC-20 (2.91 million tokens, a reduction of 53.3%), and ESC (3.31 million tokens, a reduction of 58.9%). Similarly, in the algebra subcategory, our method obtains an accuracy of 0.498 with a token consumption of 2.35 million tokens, which is comparable to ESC (0.502 accuracy, 5.69 million tokens) but with a significantly lower computational cost, reducing token consumption by 58.7%. However, in certain subcategories, such as precalculus, our method exhibits a slight drop in accuracy, achieving 0.196 compared to 0.243 for SC-10, 0.257 for SC-20, and 0.254 for ESC. This performance gap can be attributed to

	MATH-algebra		MATH-prealgebra		MATH-intermediate algebra		MATH-average	
	Accuracy	Tokens	Accuracy	Tokens	Accuracy	Tokens	Accuracy	Tokens
COT	0.364	**0.37m**	0.332	**0.23m**	0.133	**0.53m**	0.237	**0.32m**
SC-10	0.463	3.71m	0.441	2.27m	0.178	5.18m	0.315	3.13m
SC-20	**0.499**	7.47m	0.468	4.56m	**0.191**	10.43m	**0.344**	6.29m
ESC	**0.502**	5.69m	**0.474**	3.63m	**0.192**	9.69m	**0.343**	5.38m
Ours	0.498	**2.35m**	**0.573**	**1.38m**	0.151	**3.58m**	0.335	**2.07m**

Table 2: Performance evaluation of the proposed method against state-of-the-art methods on MATH-algebra, MATH-prealgebra, MATH-intermediate algebra, and MATH-average. The best performance is highlighted in bold red, and the second-best performance is highlighted in bold blue.

the insufficient exploitation of prior knowledge for this subcategory, which affects the reliability of the counterexample-based evaluation employed by our method.

Compared to ESC, our method demonstrates a more substantial reduction in token consumption while achieving comparable or better accuracy across most subcategories. Although ESC reduces token consumption by terminating the sampling process early via a sliding-window mechanism, it still fundamentally relies on repeated sampling. In contrast, our method leverages prior knowledge from the action knowledge space, enabling it to achieve similar or better performance with significantly fewer tokens. For instance, in the probability subcategory, ESC consumes 3.31 million tokens, while our method reduces this to 1.36 million tokens, a reduction of 58.9%.

Overall, the experimental results demonstrate that our method effectively balances accuracy and computational efficiency. By leveraging prior knowledge, our approach achieves significant reductions in token consumption while maintaining competitive accuracy across diverse datasets and subcategories. Compared to SC-10, our method not only achieves higher accuracy but also requires far fewer tokens. When compared to SC-20, our method achieves nearly the same accuracy with just one-third of the token consumption. Furthermore, compared to ESC, our method achieves more substantial reductions in token consumption, thanks to its ability to incorporate prior knowledge. This makes it a highly efficient solution for reasoning tasks, offering a promising trade-off between performance and computational cost.

4.5 Ablation Studies

To evaluate the effectiveness of the contradiction component in the proposed method, we conduct ablation studies by comparing three models: (1) the baseline chain-of-thought (CoT) reasoning model; (2) our model without the contradiction component; and (3) our full model. The results, shown in Table 3, demonstrate that the contradiction component improves performance by 10% for MATH-algebra and 3.5% for GSM8K. These findings highlight the significance of the contradiction-based reasoning mechanism in enhancing the reasoning capabilities of our model.

Models	GSM8k	MATH-algebra
COT	0.774	0.364
w/o Contradiction	0.803	0.393
Our full model	0.838	0.498

Table 3: Ablation study on the component of the proposed method.

5 Conclusions

In this paper, we propose a novel method for task reasoning in large language models (LLMs) through the design of an action prototype space, which efficiently organizes problem-solving strategies using unsupervised clustering and contrastive learning. Our method significantly reduces the need for repetitive computations, requiring only a single pass to achieve high-quality results, unlike traditional methods that rely on multiple iterations. By incorporating a contradiction-based answer evaluation mechanism and an action-matching inference mechanism, our approach enhances reasoning accuracy while dramatically reducing token consumption. Extensive experiments across three datasets demonstrate that our method outperforms self-consistency approaches, achieving a reduction of up to 68.5% in token usage while maintaining strong reasoning capabilities. This work highlights the potential of leveraging structured prior knowledge to optimize the efficiency and effectiveness of LLM reasoning, making it more computationally viable and aligned with human-like problem-solving strategies.

Limitations

First, our method was only tested on a limited number of datasets, and its generalizability to more diverse or complex datasets remains to be validated. Second, while our approach significantly reduces token consumption, its performance in certain scenarios is indeed inferior to results obtained through extensive iterative sampling, which may limit its effectiveness in specific complex tasks. Finally, the exploration of prior knowledge related to action prototypes can be further improved, and future work could focus on developing more efficient methods to enhance the discovery and utilization of such knowledge.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Shourya Aggarwal, Divyanshu Mandowara, Vishwajeet Agrawal, Dinesh Khandelwal, Parag Singla, and Dinesh Garg. 2021. Explanations for commonsenseqa: New dataset and models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3050–3065.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Zhaorun Chen, Zhuokai Zhao, Zhihong Zhu, Ruiqi Zhang, Xiang Li, Bhiksha Raj, and Huaxiu Yao. 2024. Autoprm: Automating procedural supervision for multi-step reasoning via controllable question decomposition. *arXiv preprint arXiv:2402.11452*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Dheeru Dua, Shivanshu Gupta, Sameer Singh, and Matt Gardner. 2022. Successive prompting for decomposing complex questions. *arXiv preprint arXiv:2212.04092*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Ellen Jiang, Kristen Olson, Edwin Toh, Alejandra Molina, Aaron Donsbach, Michael Terry, and Carrie J Cai. 2022. Promptmaker: Prompt-based prototyping with large language models. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, pages 1–8.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*.

Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. 2020. Contrastive representation learning: A framework and review. *Ieee Access*, 8:193907–193934.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. 2024. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. *arXiv preprint arXiv:2401.10480*.

Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. 2003. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

AI Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date. *Meta AI*.

Ning Miao, Yee Whye Teh, and Tom Rainforth. 2023. Selfcheck: Using llms to zero-shot check their own step-by-step reasoning. *arXiv preprint arXiv:2308.00436*.

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

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*.

Ethan Perez, Patrick Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. 2020. Unsupervised question decomposition for question answering. *arXiv preprint arXiv:2002.09758*.

Ansh Radhakrishnan, Karina Nguyen, Anna Chen, Carol Chen, Carson Denison, Danny Hernandez, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, and 1 others. 2023. Question decomposition improves the faithfulness of model-generated reasoning. *arXiv preprint arXiv:2307.11768*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2022. Distilling reasoning capabilities into smaller language models. *arXiv preprint arXiv:2212.00193*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota. Association for Computational Linguistics.

Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. 2020. What makes for good views for contrastive learning? *Advances in neural information processing systems*, 33:6827–6839.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Xinyi Wang, Alfonso Amayuelas, Kexun Zhang, Liangming Pan, Wenhu Chen, and William Yang Wang. 2024. Understanding the reasoning ability of language models from the perspective of reasoning paths aggregation. *arXiv preprint arXiv:2402.03268*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Shuang Wu, Liwen Zhu, Tao Yang, Shiwei Xu, Qiang Fu, Yang Wei, and Haobo Fu. 2024. Enhance reasoning for large language models in the game werewolf. *arXiv preprint arXiv:2402.02330*.

Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. 2024. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*.

Joshua C Yang, Marcin Korecki, Damian Dailisan, Carina I Hausladen, and Dirk Helbing. 2024. Llm voting: Human choices and ai collective decision making. *arXiv preprint arXiv:2402.01766*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

ShunYi Yeo, Gionnieve Lim, Jie Gao, Weiyu Zhang, and Simon Tangi Perrault. 2024. Help me reflect: Leveraging self-reflection interface nudges to enhance deliberativeness on online deliberation platforms. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–32.

Zhongzhi Yu, Zheng Wang, Yuhan Li, Ruijie Gao, Xiaoya Zhou, Sreenidhi Reddy Bommu, Yang Zhao, and Yingyan Lin. 2024. Edge-llm: Enabling efficient

large language model adaptation on edge devices via unified compression and adaptive layer voting. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, pages 1–6.

Yi Zhang, Sujay Kumar Jauhar, Julia Kiseleva, Ryen White, and Dan Roth. 2021. Learning to decompose and organize complex tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2726–2735.

A Prompt 1: Step-by-Step Mathematical Problem Solving

Profile: Language: English

Description: You need to answer some math questions. Please answer the given questions step by step and ultimately provide the answers to those questions. In each response, I will provide a description of the mathematical methods related to the problem, and you should use both the question itself and this description to formulate your answers. We hope that these mathematical method descriptions will help you better solve the problem rather than mislead you, so we encourage you to extract useful information from them to address the issue effectively.

Constraint: The answer must include `boxed{number}`.

Output Example: First we try factoring the left side to simplify it:

$$\frac{4x-12}{x^2+2x-15} = \frac{4(x-3)}{(x-3)(x+5)} = \frac{4}{x+5}.$$

Now we can multiply both sides by $(x+5)$ and solve for x:

$$\begin{aligned} \frac{4}{x+5} &= x+2 \quad \Rightarrow \\ 4 &= (x+5)(x+2) \quad \Rightarrow \\ &= x^2+7x+10 \quad \Rightarrow \\ 0 &= x^2+7x+6 \quad \Rightarrow \\ &= (x+6)(x+1). \end{aligned}$$

So $p=-1$ and $q=-6$, making $p-q=\boxed{5}$.

The cube root of 1000 is 10; the cube root of any number smaller than 1000 is less than 10. So, the whole numbers from 1 to 999 are the only positive whole numbers with cube roots less than 10. There are $\boxed{999}$ such numbers.

Annie and Barbara will be meeting up at the midpoint of $(6,-20)$ and $(1,14)$. We only need to find the y-coordinate of the midpoint since the problem says they only walk upwards from the midpoint to get to Charlie's location. (If you want, you can verify that the x-coordinate of the midpoint equals $7/2$.) The y-coordinate of the midpoint is $\frac{-20+14}{2} = -3$. To get to Charlie at $y=2$, the girls walk $2-(-3) = \boxed{5}$ units upward.

B Prompt 2: Evaluation of Mathematical Answers

Profile: Language: English

Description: I will give you a math problem along with an answer. Please evaluate whether the provided answer is correct based on the problem's context. If the answer logically fits the question, then it is correct; if it contradicts the description in the question, then it is incorrect. If you believe this answer is correct, you should first provide suitable reasons. Only when the reasons are sufficiently convincing should you make a judgment; otherwise, do not trust certain answers too readily. Your final output can include your thought process, but please conclude with `#### true` or `#### false`.

Constraint: Please pay attention to whether your final output includes ####. If it does not, it is considered a formatting error. Note that the final output should not just be `#### true` or `#### false`; you should write the reasons you believe the answer is correct or incorrect beforehand.

Output Example: To verify whether the answer 77 is correct, we will substitute it into the conditions of the problem and look for contradictions. The problem states that four positive integers satisfy the following conditions: 1. $a^3 = b^2$ 2. $c^5 = d^4$ 3. $c - a = 77$

The answer we are checking is $d - b = 77$. Assuming $d - b = 77$, this can be expressed as:

$$d = b + 77.$$

From the first condition $a^3 = b^2$, we can derive that $b = a^{3/2}$. Therefore,

$$d = a^{3/2} + 77.$$

Now consider the second condition $c^5 = d^4$. Substituting d into this equation gives:

$$c^5 = (a^{3/2} + 77)^4.$$

Meanwhile, the third condition states that $c = a + 77$. Substituting $c = a + 77$ into $c^5 = d^4$ yields:

$$(a+77)^5 = (a^{3/2} + 77)^4.$$

Now, we need to look for contradictions in this equation. Clearly, the two sides represent different forms. The left-hand side is a polynomial of degree five in a, while the right-hand side is a polynomial of degree four. Thus, the answer is `#### false`.