LLM Inductive Reasoning Through Multi-Agent Enhanced Monte Carlo Tree Search

Anonymous ACL submission

Abstract

Recent research reveals challenges in the inductive reasoning capabilities of large language models (LLMs). To address this challenge, we propose MATSIR, a test-time framework combining **M**ulti-**A**gents and Monte Carlo **T**ree **S**earch to strengthen LLM **I**nductive **R**easoning. MATSIR coordinates three agents (Inductor, Deductor, and Judge) that iteratively generate, validate, and refine hypotheses in a tree-search process, incorporating Deduction Reward and Judgment Reward to guide the reasoning process. Experiments on four models and five benchmarks show that MATSIR outperforms prior methods, improving Deepseek-V3 by +15.8% on average. Our analysis reveals that MATSIR reduces the hypothesis space, converges toward optimal solutions, and leverages complementary rewards of the three agents to automatically search for a better refinement path, which leads to an obvious improvement of the inductive reasoning capability of LLMs. MATSIR's code is fully open-sourced at https://anonymous.4open.science/r/MATSIR-D6C5.

1 Introduction

Inductive reasoning (i.e., Induction) is considered a fundamental reasoning approach (Singmann and Klauer, 2011; Evans and Over, 2013). Induction transforms specific observations into formal theories. It is the cornerstone of scientific discovery that allows patterns to emerge from raw data (Govier, 1997). It serves as a foundational principle of the scientific methods (Hepburn and Andersen, 2021; Stadler, 2004; Lawson, 2005), guiding our understanding of the world. Figure 1 succinctly illustrates how the law of universal gravitation was discovered through inductive reasoning. Naturally, induction is regarded as a fundamental aspect of intelligence (Peirce, 1868).

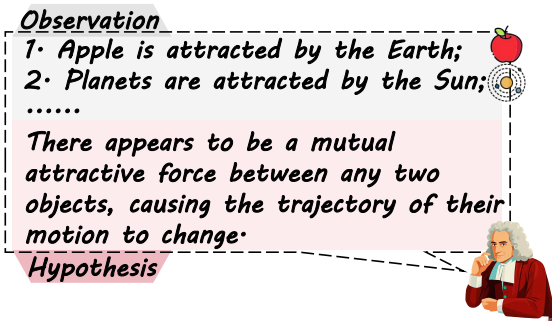

Figure 1: A concise example of human induction.

Recently, as the capabilities of large language models have been continuously strengthened (Wei et al., 2022a; Qin et al., 2024), researchers have also begun to focus on whether large language models can perform logical thinking like humans (Wei et al., 2022b; Yasunaga et al., 2024; Yuan et al., 2024; Kojima et al., 2022; Qin et al., 2024; YAO et al., 2024). One important criterion for evaluation is inductive reasoning.

Previous studies have extensively investigated the inductive reasoning capacities of pre-trained large language models (LLMs), primarily employing input-output (IO) prompting methodologies (Gendron et al., 2024; Yang et al., 2024; Moskvichev et al., 2023; Mirchandani et al., 2023; Tang et al., 2023; Xu et al., 2025; Han et al., 2024; Xu et al., 2024a; Webb et al., 2023; Wang et al., 2024b; Qiu et al., 2024b; Sun et al., 2024). Specifically, an induction problem consists of two components: demonstration samples ($\mathcal{D}$) and test samples ($\mathcal{T}$), both are structured as input-output pairs. Usually, the demonstration samples include multiple IO pairs ($\mathcal{D} = \{(i_1, o_1), (i_2, o_2), \cdots\}$), whereas the test sample comprises one single IO pair($\mathcal{T} = \{(i_t, o_t)\}$). Within this induction problem, there exists a mapping function f between input and output such that $o = f(i)$. The primary objective of a LLM inductor is to analyze

patterns within demonstration samples $\mathcal{D}$, generate hypotheses about the correct mapping function f through inductive reasoning. Its induction capability is then evaluated by verifying the correctness of the hypothesis on $\mathcal{T}$. This problem formulation closely resembles In-Context Learning (ICL) (Dong et al., 2024), but differs in two key aspects: 1) ICL does not require the model to identify the correct mapping function f, but only to produce the correct test output o_t; 2) ICL tasks rely on the model's world knowledge, whereas induction problems deliberately remove this factor to focus on pure inductive reasoning capabilities (Chollet, 2019; Qiu et al., 2024b).

However, existing research indicates that the inductive reasoning capabilities of large language models remain significantly inferior to their deductive reasoning capabilities (Gendron et al., 2024; Xu et al., 2025). To enhance the inductive reasoning capabilities of large language models, we have devised a plug-and-play pipeline applicable at test-time, which integrates multi-agent coordination with Monte Carlo Tree Search. To summarize, our contributions are as follows:

- We propose **MATSIR**, a test-time multi-agent Monte Carlo Tree Search framework that enhances inductive reasoning in large language models without additional fine-tuning.
- We design a dual-reward mechanism that guides reasoning toward logically coherent and semantically accurate hypotheses.
- Through detailed analyses, we find that MATSIR effectively not only prunes the hypothesis space, but also automatically searches for a better refinement path.
- We show that MATSIR generalizes across diverse LLM architectures and scales, providing a plug-and-play approach for strengthening inductive reasoning.

2 Related Work

2.1 Inductive Reasoning for LLM

Considering that hypothesis generation constitutes the primary goal of LLM inductors, targeted hypothesis refinement at test-time becomes a particularly efficient solution. Qiu et al. (2024b) systematically investigated the inductive reasoning capabilities of language models through iterative hypothesis refinement, revealing their dual nature as both proficient hypothesis proposers and puzzlingly inconsistent reasoners when compared to human cognition. To further advance the inductive reasoning capabilities of LLMs, Wang et al. (2024b) proposed a structured pipeline that systematically generates natural language hypotheses, translates them into verifiable Python programs, and selects the most generalizable solutions. Our approach builds upon comparable theoretical principles(test-time computing & hypothesis refinement) while achieving superior architectural elegance in pipeline implementation. Sun et al. (2024) focused on a fine-tuning approach on open-sourced small LLMs; they propose a deductive reasoning-based method for automated training data generation, which effectively improved the model's inductive reasoning performance.

2.2 LLM-Based Multi-Agent System

A multi-agent system based on LLMs is an AI framework where multiple specialized agents, each assigned with distinct functional roles (e.g., analyst, executor), collaborate through the core capabilities of LLMs (Guo et al., 2024). This system achieves complex task completion through coordinated division of labor (Khot et al., 2023; Hong et al., 2024), dynamic interactions (Liu et al., 2023), and tool utilization (Li et al., 2023; Ruan et al., 2023), effectively emulating human teamwork by decomposing problems into sub-tasks handled by different specialist agents to generate the final output. Such systems demonstrate significant potential in automated workflows, multi-perspective decision-making, and social simulation experiments (Park et al., 2023; Gao et al., 2023; Chen et al., 2024).

2.3 Tree-Search Algorithm Applied to LLM

Feng et al. (2023) highlighted the viability of tree-search algorithms for boosting LLMs' reasoning abilities. Ma et al. (2024) demonstrated that step-level reward models (SRMs) effectively enhance mathematical reasoning through process supervision or reinforcement learning-based alignment, with Monte Carlo Tree Search (MCTS) proving high effectiveness for automated preference annotation. Conversely, some researchers suggested that even test-time-only tree-search reasoning pipelines (without additional training) still substantially enhance LLMs' mathematical problem-solving capabilities (Zhang et al., 2024a; Wang et al., 2024a; Zhang et al., 2024b). Although tree-

search algorithms have demonstrated efficacy in mathematical problem-solving, their potential for abstract logical reasoning tasks remains largely unexplored and understudied.

3 Methodology

3.1 Components

1) Modified Tree-Search Algorithm for Induction Problems. To improve the integration of the tree-search strategy with the multi-agent approach for solving inductive reasoning problems, several modifications have been made to the Monte Carlo Tree Search (MCTS).

- **Rearranging Demonstrations**: For each problem, the multiple demonstrations are reordered. By exploring various permutations of these demonstrations, the inductive process becomes more robust, leading to more generalized and accurate hypotheses.
- **Reserving a Demonstration for Validation**: The last one of the Rearranged demonstrations is reserved as a validation sample. This allows **Deductor** to employ deductive reasoning to verify the correctness of generated hypotheses to provide Deduction Reward.
- **Customized UCT**: Our approach innovatively utilizes both **Deductor** and **Judge** to score hypotheses, introducing two distinct reward sources. To accommodate this dual-reward mechanism, the Upper Confidence Bound for Trees (UCT) formula has been specifically adapted, thereby providing a more comprehensive evaluation of each hypothesis and guiding the tree-search process more effectively.

2) Multi-Agent based Reasoning and Rewarding. Within MATSIR, three LLM agents are purposefully integrated into the tree-search process, as depicted in Figure 2

- **Inductor** agent performs inductive reasoning on the given examples to generate hypothesis or refine existing hypothesis. The quality of the hypothesis is critical, as it forms the basis for solving the reasoning problem.
- **Deductor** agent applies the hypothesis inferred by **Inductor** to a new input to generate an output. Additionally, if the input originates from a verification sample, the Inductor uses a tool to assess whether the inferred output matches the expected output of the verification sample. This comparison yields a score, which functions as the Deduction Reward.
- **Judge** agent evaluates the induction process carried out by **Inductor**, assessing the logical consistency, coherence, and overall quality of the reasoning. Based on this evaluation, **Judge** assigns a score that acts as an additional reward. This score also plays a crucial role in refining the inductive reasoning process in subsequent iterations (referred to as Judgment Reward).

3.2 Reasoning Process

For each problem, MATSIR employs a corresponding tree search process, as illustrated in subplot (a) on the left of Figure 2. To clearly visualize the tree-structured nature of MATSIR, we adopt a programming-style notation in the following formal descriptions for intuitive representation. Here, n denotes a node in the tree, and $n.*$ refers to an attribute of that node (e.g., n.parent represents the parent node of n).

1) Variation. The Variation phase ensures broader exploration of the sample space, thereby enhancing diversity. When receiving a problem, MATSIR initially establishes a root node. The demonstrations $\mathcal{D} = \{(i_1, o_1), (i_2, o_2), \cdots, (i_m, o_m)\}$ associated with the problem (assuming there are m demonstrations in one problem) are then rearranged, yielding $m!$ possible permutations.

$$\mathcal{P} = \text{Permutations}(\mathcal{D}), \quad |\mathcal{P}| = m! \tag{1}$$

To optimize computational resources and time, a maximum number of child nodes is imposed on the tree nodes. Therefore, if the number of permutations exceeds this maximum number, a subset of permutations, equal to the permitted maximum number of child nodes$N_{\max}$, is randomly selected.

$$\mathcal{C} = \begin{cases} \mathcal{P} & \text{if } m! \leq N_{\max}, \\ \text{RandomSubset}(\mathcal{P}, N_{\max}) & \text{if } m! > N_{\max}. \end{cases} \tag{2}$$

Here, $\text{RandomSubset}(\mathcal{P}, N_{\max})$ selects $N_{\max}$ distinct permutations uniformly at random from $\mathcal{P}$.

For each candidate child node, 1) **Inductor** derives a hypothesis (H) with its inductive reasoning process (I) by applying inductive reasoning to the contained demonstrations **without the last pair held-out as validation sample** (i_v, o_v), as shown in Equation 3; 2) **Deductor** performs deductive reasoning on the input of the verification

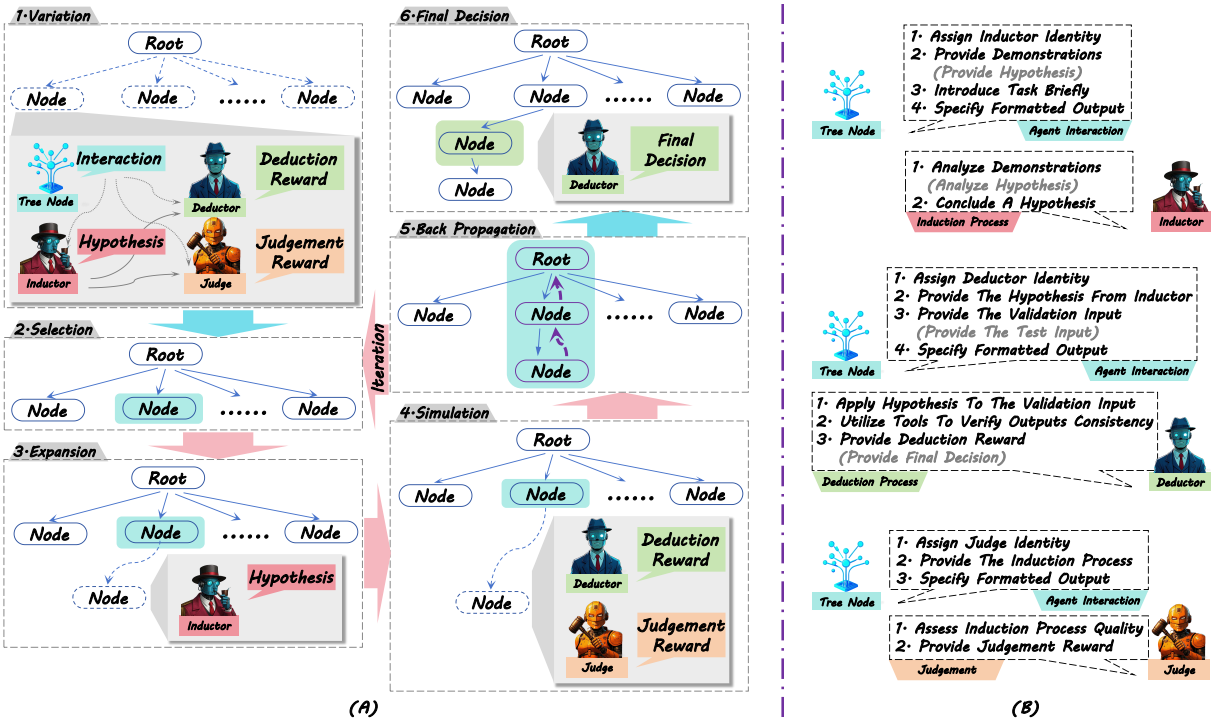

Figure 2: Overview of the MATSIR framework. (A) shows the customized Monte Carlo tree search process for inductive reasoning, where each node represents a hypothesis derived from rearranged demonstrations. The search explores and refines hypotheses through Variation, Selection, Expansion, Simulation, and Back Propagation guided by dual rewards. (B) illustrates the roles of the three agents: **Inductor** generates hypotheses, **Deductor** verifies them using reserved samples to produce Deduction Rewards, and **Judge** evaluates reasoning quality to provide Judgment Rewards.

sample (i_v, o_v) using the hypothesis generated by **Inductor** (H), and subsequently employs a tool to compare the output derived from this deductive reasoning with the output of the verification sample. This comparison yields a score, referred to as the Deduction Reward (Q_d), as shown in Equation 4; 3) **Judge** evaluates the induction process (I) conducted by the inductor, thereby assigning a score known as the Judgment Reward(Q_j), as shown in Equation 5.

$$n.H, n.I = \textbf{\textit{Inductor}}\left((i_1, o_1), (i_2, o_2), \cdots\right) \quad (3)$$

$$n.Q_d = \textbf{\textit{Deductor}}\left(n.H, (i_v, o_v)\right) \quad (4)$$

$$n.Q_j = \textbf{\textit{Judge}}\left(n.I\right) \quad (5)$$

2) Selection. The selection phase in MATSIR, akin to that in Monte Carlo Tree Search (MCTS), is a pivotal step that strikes a balance between exploration and exploitation. In this phase, the algorithm navigates from the root to a leaf node by selecting child nodes based on the Upper Confidence Bound for Trees (UCT) criterion. The UCT value integrates the node's value estimate with an exploration term, thereby fostering the selection of promising nodes while also encouraging the exploration of less frequently visited ones. MATSIR has two distinct reward signals: one from the Deductor and the other from the Judge. To accommodate this, we have refined the UCT formula to incorporate both rewards, shown as follows:

$$UCT(n) = a{\cdot}n.Q_d + b{\cdot}n.Q_j + c{\cdot}\sqrt{\frac{2\ln V(n.parent)}{V(n)}} \quad (6)$$

where $n.Q_d$ is the Deduction Reward obtained from node n; $n.Q_j$ is the Judement Reward obtained from node n; $V(n)$ is the number of times node n has been visited; $n.parent$ is the parent node of node n; a and b are weighting factors that balance the influence of the two rewards and $a + b = 1.0$; c is the exploration parameter.This enhanced UCT formula allows our framework to effectively balance exploration and exploitation while incorporating multiple sources of feedback to improve the reasoning process.

3) Expansion. The expansion phase is part of the algorithm for exploring and expanding the tree nodes. Firstly, based on the nodes demonstrations and hypothesis, it employs the **Inductor** to rethink

Algorithm 1 Update Q-values

Input: n (current node)
Output: Updated values for all parent nodes
Function `update_Q-values(`n`)`**:**
  $p \leftarrow n$.parent
  while $p \neq \emptyset$ **do**
    $\hat{Q}_d \leftarrow \max_{c \in p.\text{children}} c.Q_d$
    $p.Q_d \leftarrow \frac{p.Q_d + \hat{Q}_d}{2}$
    $\hat{Q}_j \leftarrow \max_{c \in p.\text{children}} c.Q_j$
    $p.Q_j \leftarrow \frac{p.Q_j + \hat{Q}_j}{2}$
    $p \leftarrow p$.parent

Algorithm 2 Find Best Hypothesis

Input: n_{root}(Root node of the tree)
Output: The node with best hypothesis on the tree
Function `find_best_hyperthesis(`n_{root}`)`**:**
  $n_h \leftarrow$ **Traverse**(n_{root})
  return $n_h.H$
Function `Traverse(`n`)`**:**
  if $n.children = \emptyset$ **then**
    return n
  else
    $n \leftarrow \arg\max_{c \in n.children}$ **SumQ(Traverse(**c**))**
    return n
Function `SumQ(`n`)`**:**
  return $a * n.Q_d + b * n.Q_j$

the hypothesis by generating a new induction process and hypothesis.

$$n.H, n.I = \textbf{\textit{Inductor}}(n.parent.H) \quad (7)$$

This is followed by the creation of a new child node. Meanwhile, it checks whether the current node has reached its maximum allowed number of child nodes, or if other conditions warrant the cessation of further expansion. If such conditions are met, the node is marked as fully expanded. In general, the expansion phase adds a new child node with a novel hypothesis to the selected node from the selection phase.

4) Simulation. The simulation scores the hypotheses within the newly generated child nodes obtained through the expansion phase. **Deductor** evaluates the hypothesis by comparing its derived output against the verification samples ground truth, computing the Deduction Reward (Equation 4), while **Judge** assesses the Inductors reasoning process to provide the Judgment Reward (Equation 5).

5) Back Propagation. The back propagation phase is responsible for updating the Q-values of nodes in MATSIR, starting from a given node on the tree and moving upwards to its ancestors (as shown in Algorithm 1). For each parent node encountered, a function calculates the maximum Q_j and Q_d values among its children. It then computes the average of these maximum values with the current Q_j and Q_d the values of its parent node. The parent node's Q_j and Q_d values are subsequently updated with these new averages. This process continues recursively up the tree until the root node is reached, ensuring that the Q-values reflect the best possible outcomes from each node's perspective.

6) Final Decision. To identify the best hypothesis within a tree structure, Algorithm 2 is designed to traverse the tree and evaluate the Q-values of the nodes. This traversal is performed by a nested function **Traverse**, which recursively explores each node's children and selects the child with the highest Q-value, as determined by the **SumQ** function. This process continues until a leaf node is reached, which represents the best hypothesis node(n_h). Finally, **Deductor** performs deductive reasoning on the input of test sample(i_t) based on this hypothesis($n_h.H$) to obtain the final answer ($\hat{o}_t$):

$$\hat{o}_t = \textbf{\textit{Deductor}}\,(n_h.H, i_t) \quad (8)$$

4 Experiment

4.1 Experiment Settings

We conduct our experiments using four open-sourced large language models: Deepseek-V3 (DeepSeek-AI et al., 2025b), QWQ-32B (QwenLM Team, 2023), Deepseek-R1-Distilled-Qwen-32B (R1-Qwen-32B) (DeepSeek-AI et al., 2025a), and GLM-Z1-32B (THUDM-GLM, 2025) (Details in Appendix A.1). For evaluation, we randomly select 110 samples from each of five inductive reasoning benchmarks: ListFunction (Rule, 2020), 1D-ARC (Xu et al., 2024b), ARC2024 (Chollet, 2019), MiniARC (Kim et al., 2022), and ConceptARC (Moskvichev et al., 2023) (Details in Appendix A.2). We compare four baseline methods with MATSIR: Simple Transduction, Self-Consistency (SC) (Wang et al., 2023), Self-Refine (SR) (Madaan et al., 2023) and Hypothesis Refine (HR) (Qiu et al., 2024a) (Details in Appendix A.3). Hyperparameters are shown in Appendix A.4.

Model	Method	Task					AVG.
		ListFunction	ARC2024	1D-ARC	MiniARC	ConceptARC	
Deepseek-V3	Simple Transduction	63.6	11.7	63.4	20.1	25.3	36.8
	SC [ICLR2023] (Wang et al., 2023)	35.4	12.7	7.3	3.6	20.0	15.8
	SR [NIPS2023] (Madaan et al., 2023)	35.4	1.8	8.2	7.3	16.4	13.8
	HR [ICLR2024] (Qiu et al., 2024a)	77.3	5.4	64.5	18.2	25.4	38.2
	MATSIR [OURS]	**87.3**	**30.9**	**89.1**	**31.8**	**30.9**	**54.0**
	(Progress)	(↑ 10.0)	(↑ 18.2)	(↑ 24.6)	(↑ 11.7)	(↑ 5.5)	(↑ 15.8)
QWQ-32B	Simple Transduction	63.6	1.8	50.9	13.6	7.3	27.4
	SC [ICLR2023] (Wang et al., 2023)	23.6	9.1	16.4	23.6	20.0	18.5
	SR [NIPS2023] (Madaan et al., 2023)	27.3	1.8	7.3	27.3	1.8	13.1
	HR [ICLR2024] (Qiu et al., 2024a)	63.6	5.4	61.8	18.2	25.4	34.9
	MATSIR [OURS]	**91.5**	**5.5**	**61.8**	**39.6**	**16.2**	**42.9**
	(Progress)	(↑ 27.9)	(↓ 3.6)	(↑ 0.0)	(↑ 12.3)	(↓ 9.2)	(↑ 8.0)
R1-Qwen-32B	Simple Transduction	69.2	6.5	46.5	20.1	11.8	30.8
	SC [ICLR2023] (Wang et al., 2023)	27.3	9.1	12.7	3.6	18.2	14.2
	SR [NIPS2023] (Madaan et al., 2023)	20.0	1.8	7.3	7.3	1.8	7.6
	HR [ICLR2024] (Qiu et al., 2024a)	63.6	5.4	61.8	18.2	25.4	34.9
	MATSIR [OURS]	**86.6**	**4.5**	**57.3**	**23.6**	**13.0**	**37.0**
	(Progress)	(↑ 17.4)	(↓ 4.6)	(↓ 4.5)	(↑ 3.5)	(↓ 12.4)	(↑ 2.1)
GLM-Z1-32B	Simple Transduction	58.2	3.6	42.7	12.7	5.4	24.5
	SC [ICLR2023] (Wang et al., 2023)	37.3	9.1	14.5	0.0	18.2	15.8
	SR [NIPS2023] (Madaan et al., 2023)	35.4	1.8	7.3	7.3	1.8	10.7
	HR [ICLR2024] (Qiu et al., 2024a)	77.3	5.4	61.8	18.2	25.4	37.6
	MATSIR [OURS]	**79.5**	**6.4**	**62.7**	**38.2**	**16.4**	**40.6**
	(Progress)	(↑ 2.2)	(↓ 2.7)	(↑ 0.9)	(↑ 20.0)	(↓ 9.0)	(↑ 3.0)

Table 1: Performance of different methods employing 4 models across 5 different tasks. Scores achieved by MATSIR are shaded in gray . Underlined values indicate the highest scores among existing baseline methods for each task. *(Progress)* denotes performance change of MATSIR compared to the current highest score, with (↑ *green*) representing improvement and (↓ *red*) indicating regression.

4.2 Main Results

The experimental results demonstrate that MATSIR achieves superior performance in most scenarios, as shown in Table 1. Specifically, when implemented on Deepseek-V3, our method establishes new state-of-the-art performance across all five tasks, with remarkable improvements of +10.0% on ListFunction, +18.2% on ARC2024, +24.6% on 1D-ARC, +11.7% on MiniARC, and +5.5% on ConceptARC. This leads to an average performance gain of +15.8% over the strongest baseline (Hypothesis Refine).

For other model architectures, MATSIR maintains competitive performance while showing varying degrees of improvement. For QWQ-32B, MATSIR achieves significant gains in ListFunction (+27.9%) and MiniARC (+12.3%), although it shows slight regression on ARC2024 (-3.6%) and ConceptARC (-9.2%). For Deepseek-R1-Distilled-Qwen-32B, our method demonstrates strong performance on ListFunction (+17.4%) while maintaining comparable results on other tasks. For GLM-Z1-32B, MATSIR shows particularly impressive improvement on MiniARC (+20.0%) while maintaining stable performance on other tasks.

Generally, MATSIR consistently outperforms existing methods in terms of average performance across all model architectures, demonstrating its robustness and generalizability. Also, MATSIR achieves the most significant improvements on Deepseek-V3, suggesting that the effectiveness of our method may be influenced by the base model's capability levels.

4.3 Analysis Study

Analysis 1: Does MATSIR automatically search towards the correct solution? During the search process within MATSIR, **Inductor** refines hypotheses provided by previous inductive reasoning processes. A pertinent question herein is whether this optimization within the hypothesis space converges towards the optimal hypothesis, thereby enhancing its validity and reducing reasoning costs?

Figure 3 illustrate the trends in node quantity and reward value respectively, as the search tree progressively deepens (with level 0 representing the root). As shown in its upper part, the num-

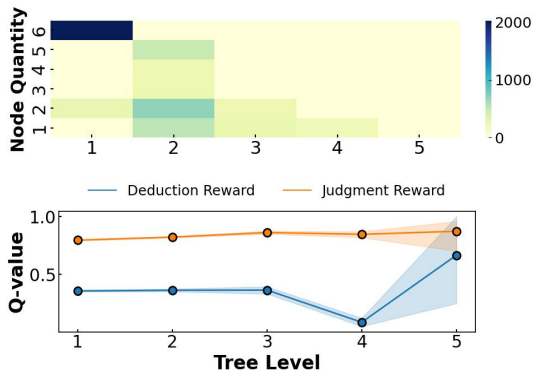

Figure 3: Node quantity (upper) and average values of 2 rewards (lower) as search trees deepen.

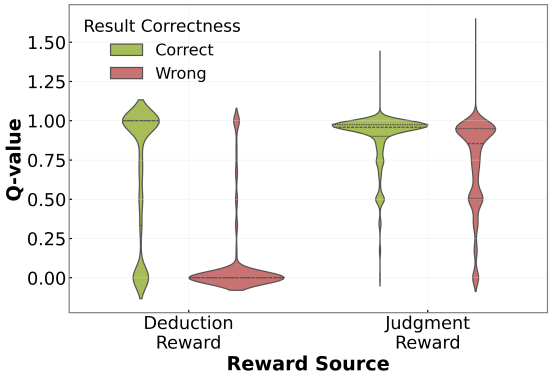

Figure 4: Distributive patterns of different rewards on best-hypotheses-nodes and their association with final results correctness

ber of nodes decreases progressively as the tree depth increases. This indicates that the search tree becomes increasingly sparse under the guide of our method, suggesting a preference for exploring fewer but more promising branches rather than exhaustively traversing the entire hypothesis space. Lower part of Figure 3 demonstrates that both reward metrics generally exhibit an upward trend as the tree depth increases, indicating that the search process progressively optimizes the hypotheses.

Figure 9 in Appendix A.7 illustrates that MATSIR possesses a certain level of error-correction capability. Instead of persistently following a single search path, it can automatically discover a better refinement path, leading to a better hypothesis.

Finding 1: MATSIR guides LLMs improve inductive reasoning process while also is able to trace back and correct errors.

Analysis 2: Which rewarding strategy carries greater significance? Existing test-time scaling methods tend to employ LLMs-as-Judges to refine model responses. But verifiable reward methods also have shown its effectiveness. We design two distinct reward sources for MATSIR, enabling a comparative evaluation of **Judge**'s effectiveness.

Figure 4 displays the distribution of two key reward metrics (Deduction Reward and Judgment Reward) from the best-hypothesis nodes in Table 1's experimental results. Ideally, correct answers should exhibit reward values near 1, while incorrect ones should approach 0. Our results reveal a clear distinction between the two reward mechanisms: while Deduction Reward closely follows the ideal trend, Judgment Reward exhibits a substantial divergence. Specifically, Judgment Reward demonstrates a persistent bias toward high

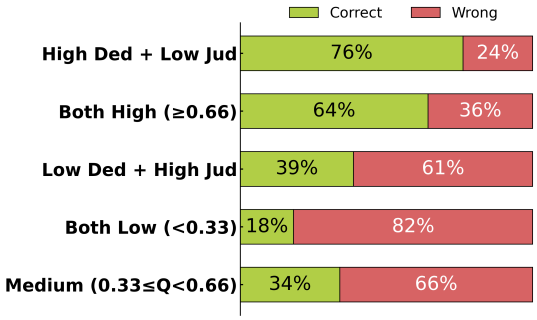

Figure 5: Proportion of correct and wrong responses across five conditions.

values, even in cases where the final result is wrong. Figure 7 in Appendix A.6 presents a fine-grained breakdown of the reward distribution, revealing an obvious limitation of Judgment Reward: Across all 20 experimental trials, only two cases produce scores that are not significantly wrong (subplot F and J).

Finding 2: Compared with the LLM-as-Judge paradigm, verifiable rewards performs better.

Analysis 3: To what extent are these rewarding strategies complementary or conflicting? Finding 2 indicates that the Deduction Reward is more pivotal in the MATSIR problem-solving process. Thus, does the Judgment Reward collaborate with or conflict with the Deduction Reward? Specifically, we examine: 1) When the Deduction Reward is relatively high, does a lower Judgment Reward potentially compromise MATSIR's performance by inducing wrong outputs? 2) When Deduction Reward is relatively low, does a higher Judgment Reward facilitate MATSIR in generating correct responses?

Figure 5 illustrates the distribution of response

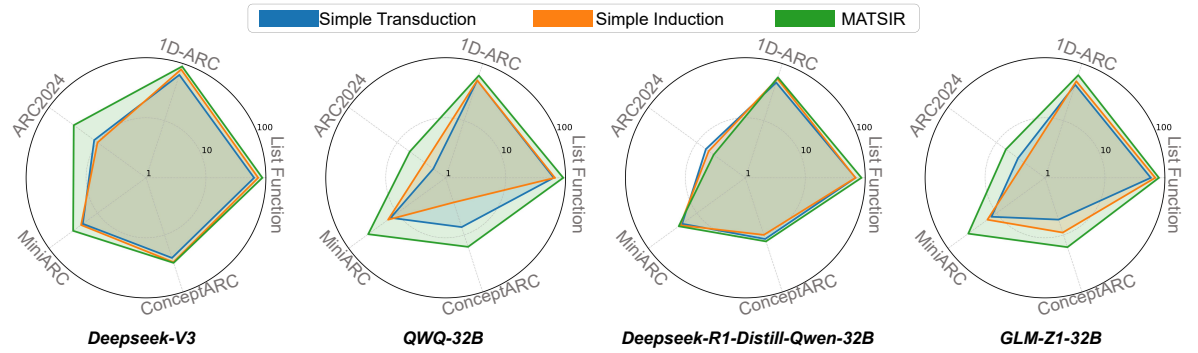

Figure 6: Comparative performance assessed on 5 different tasks, presented in **log-scale** radar plots.

correctiveness across five cognitive conditions (Low: Q-value < 0.33; Medium: 0.33<= Q-value < 0.66; High: Q-value >= 0.66), each defined by different levels or combinations of Deduction Reward (Ded) and Judgment Reward (Jud) reasoning quality. A comparison between the "Both High" and "High Deduction + Low Judgment" conditions reveals that elevated Judgment Reward exerts a negative impact on MATSIR performance when Deduction Reward is already maintained at a high level. Conversely, examination of the "Both Low" versus "Low Deduction + High Judgment" conditions demonstrates that increased Judgment Reward can effectively guide MATSIR toward correct responses when Deduction Reward remains at a lower baseline.

Finding 3: **Judge** plays a constructive role when **Deductor** performs poorly, but becomes a liability when **Deductor** performs well.

Analysis 4: Is the improvement offered by MATSIR grounded in the enhancement of the model's intrinsic inductive reasoning capabilities? To investigate this question, we design prompts that instruct the model to conclude the underlying hypothesis from the given demonstration, and subsequently apply this hypothesis to the test sample **within a single request**. This setting is referred to as Simple Induction.

As illustrated in the Figure 6, comparative analysis between Simple Induction and MATSIR demonstrates that generally MATSIR achieves consistent performance improvements across all models and tasks. These results demonstrate that MATSIR effectively enhances the fundamental induction capability of the models. MATSIR does not enhance (and even slightly degraded) the reasoning capability of Deepseek-R1-Distilled-Qwen-32B. This can be attributed to the fact that Deepseek-R1-Distilled-Qwen-32B is solely distilled through supervised fine-tuning (SFT) without subsequent preference optimization fine-tuning. In contrast, the other three models are fine-tuned by preference optimization, which equips them with superior generalization abilities and greater likelihood of successfully following instructions.

Finding 4: MATSIR is more effective in improving the inductive reasoning ability of the models fine-tuned via preference optimization than the distilled ones.

5 Conclusion

In this work, we introduce **MATSIR**, a novel framework designed to enhance the inductive reasoning capability of LLMs through the integration of multi-agents and Monte Carlo Tree Search. By leveraging the strengths of three specialized agents (**Inductor**, **Deductor**, and **Judge**), MATSIR systematically explores, validates, and refines hypotheses (proposed by **Inductor**), achieving significant improvements over existing methods. Our experimental results across diverse models and benchmarks demonstrate the framework's effectiveness in inductive reasoning tasks.

Limitation

As a test-time scaling method, MATSIR fundamentally relies on the underlying capabilities of the base LLM. If the model itself cannot generate reasonable hypotheses, the multi-agent search may fail to converge to a correct solution, limiting the achievable performance gains. Nevertheless, MATSIR is model-agnostic and can be seamlessly integrated with a wide range of LLM architectures without additional fine-tuning.

References

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2024. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The Twelfth International Conference on Learning Representations (ICLR)*.

François Chollet. 2019. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025a. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025b. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, and 1 others. 2024. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jonathan Evans and David E. Over. 2013. Reasoning to and from belief: Deduction and induction are still distinct. *Thinking & Reasoning*, 19:267 – 283.

Xidong Feng, Ziyu Wan, Muning Wen, Ying Wen, Weinan Zhang, and Jun Wang. 2023. Alphazero-like tree-search can guide large language model decoding and training. In *NeurIPS 2023 Foundation Models for Decision Making Workshop (NIPS FMDM workshop)*.

Chen Gao, Xiaochong Lan, Zhihong Lu, Jinzhu Mao, Jinghua Piao, Huandong Wang, Depeng Jin, and Yong Li. 2023. S3: Social-network simulation system with large language model-empowered agents. *arXiv preprint arXiv:2307.14984*.

Gael Gendron, Qiming Bao, Michael Witbrock, and Gillian Dobbie. 2024. Large language models are not strong abstract reasoners yet. In *ICLR 2024 Workshop: How Far Are We From AGI*.

T. Govier. 1997. *A Practical Study of Argument*. Philosophy Series. Wadsworth Publishing Company.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI)*.

Simon Jerome Han, Keith J. Ransom, Andrew Perfors, and Charles Kemp. 2024. Inductive reasoning in humans and large language models. *Cognitive Systems Research*, 83:101155.

Brian Hepburn and Hanne Andersen. 2021. Scientific Method. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*, Summer 2021 edition. Metaphysics Research Lab, Stanford University.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Kaggle. 2024. Arc prize 2024 competition data. `https://www.kaggle.com/competitions/arc-prize-2024/data`. Accessed: 2024-05-06.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations (ICLR)*.

Subin Kim, Prin Phunyaphibarn, Donghyun Ahn, and Sundong Kim. 2022. Playgrounds for abstraction and reasoning. In *NeurIPS 2022 Workshop on Neuro Causal and Symbolic AI (NIPS nCSI workshop)*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in neural information processing systems (NIPS)*.

Anton E. Lawson. 2005. What is the role of induction and deduction in reasoning and scientific inquiry. *Journal of Research in Science Teaching*, 42:716–740.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. Api-bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLLP)*.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2023. Dynamic llm-agent network: An llm-agent collaboration framework with agent team optimization. *arXiv preprint arXiv:2310.02170*.

Yiran Ma, Zui Chen, Tianqiao Liu, Mi Tian, Zhuo Liu, Zitao Liu, and Weiqi Luo. 2024. What are step-level reward models rewarding? counterintuitive findings from mcts-boosted mathematical reasoning. *arXiv preprint arXiv:2412.15904*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback.

Suvir Mirchandani, Fei Xia, Pete Florence, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, Andy Zeng, and 1 others. 2023. Large language models as general pattern machines. In *7th Annual Conference on Robot Learning (CoRL)*.

Arsenii Kirillovich Moskvichev, Victor Vikram Odouard, and Melanie Mitchell. 2023. The conceptARC benchmark: Evaluating understanding and generalization in the ARC domain. *Transactions on Machine Learning Research (TMLR)*.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology (UIST)*.

C. S. Peirce. 1868. Questions concerning certain faculties claimed for man. *The Journal of Speculative Philosophy*, 2:103–114.

Chengwei Qin, Wenhan Xia, Tan Wang, Fangkai Jiao, Yuchen Hu, Bosheng Ding, Ruirui Chen, and Shafiq Joty. 2024. Relevant or random: Can llms truly perform analogical reasoning? *arXiv preprint arXiv:2404.12728*.

Linlu Qiu, Liwei Jiang, Ximing Lu, Melanie Sclar, Valentina Pyatkin, Chandra Bhagavatula, Bailin Wang, Yoon Kim, Yejin Choi, Nouha Dziri, and Xiang Ren. 2024a. Phenomenal yet puzzling: Testing inductive reasoning capabilities of language models with hypothesis refinement. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Linlu Qiu, Liwei Jiang, Ximing Lu, Melanie Sclar, Valentina Pyatkin, Chandra Bhagavatula, Bailin Wang, Yoon Kim, Yejin Choi, Nouha Dziri, and 1 others. 2024b. Phenomenal yet puzzling: Testing inductive reasoning capabilities of language models with hypothesis refinement. In *The Twelfth International Conference on Learning Representations (ICLR)*.

QwenLM Team. 2023. Qwenlm blog - qwq-32b. https://qwenlm.github.io/blog/qwq-32b/. Accessed: 2025.05.06.

Jingqing Ruan, Yihong Chen, Bin Zhang, Zhiwei Xu, Tianpeng Bao, Guoqing Du, Shiwei Shi, Hangyu Mao, Ziyue Li, Xingyu Zeng, and 1 others. 2023. Tptu: large language model-based ai agents for task planning and tool usage. *arXiv preprint arXiv:2308.03427*.

Joshua Stewart Rule. 2020. *The child as hacker: building more human-like models of learning*. Ph.D. thesis, Massachusetts Institute of Technology.

Henrik Singmann and Karl Christoph Klauer. 2011. Deductive and inductive conditional inferences: Two modes of reasoning. *Thinking & Reasoning*, 17:247 – 281.

Friedrich Stadler. 2004. *Induction and Deduction in the Sciences*. Springer.

Wangtao Sun, Haotian Xu, Xuanqing Yu, Pei Chen, Shizhu He, Jun Zhao, and Kang Liu. 2024. Itd: Large language models can teach themselves induction through deduction. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Xiaojuan Tang, Zilong Zheng, Jiaqi Li, Fanxu Meng, Song-Chun Zhu, Yitao Liang, and Muhan Zhang. 2023. Large language models are in-context semantic reasoners rather than symbolic reasoners. *arXiv preprint arXiv:2305.14825*.

THUDM-GLM. 2025. Glm-z1 series. https://huggingface.co/THUDM/GLM-Z1-32B-0414. Accessed: 2025.05.06.

Ante Wang, Linfeng Song, Ye Tian, Baolin Peng, Dian Yu, Haitao Mi, Jinsong Su, and Dong Yu. 2024a. Litesearch: Efficacious tree search for llm. *arXiv preprint arXiv:2407.00320*.

Ruocheng Wang, Eric Zelikman, Gabriel Poesia, Yewen Pu, Nick Haber, and Noah Goodman. 2024b. Hypothesis search: Inductive reasoning with language models. In *The Twelfth International Conference on Learning Representations (ICLR)*.

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

Taylor Webb, Keith J Holyoak, and Hongjing Lu. 2023. Emergent analogical reasoning in large language models. *Nature Human Behaviour*, 7(9):1526–1541.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent abilities of large language models. *Transactions on Machine Learning Research (TMLR)*.

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

Fangzhi Xu, Qika Lin, Jiawei Han, Tianzhe Zhao, Jun Liu, and Erik Cambria. 2025. Are Large Language Models Really Good Logical Reasoners? A Comprehensive Evaluation and Beyond . *IEEE Transactions on Knowledge & Data Engineering (TKDE)*, 37:1620–1634.

Yudong Xu, Wenhao Li, Pashootan Vaezipoor, Scott Sanner, and Elias Boutros Khalil. 2024a. Llms and the abstraction and reasoning corpus: Successes, failures, and the importance of object-based representations. *Transactions on Machine Learning Research (TMLR)*.

Yudong Xu, Wenhao Li, Pashootan Vaezipoor, Scott Sanner, and Elias Boutros Khalil. 2024b. LLMs and the abstraction and reasoning corpus: Successes, failures, and the importance of object-based representations. *Transactions on Machine Learning Research (TMLR)*.

Zonglin Yang, Li Dong, Xinya Du, Hao Cheng, Erik Cambria, Xiaodong Liu, Jianfeng Gao, and Furu Wei. 2024. Language models as inductive reasoners. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (ACL)*.

Yuxuan YAO, Han Wu, Zhijiang Guo, Zhou Biyan, Jiahui Gao, Sichun Luo, Hanxu Hou, Xiaojin Fu, and Linqi Song. 2024. Learning from correctness without prompting makes LLM efficient reasoner. In *First Conference on Language Modeling (CoLM)*.

Michihiro Yasunaga, Xinyun Chen, Yujia Li, Panupong Pasupat, Jure Leskovec, Percy Liang, Ed H. Chi, and Denny Zhou. 2024. Large language models as analogical reasoners. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Siyu Yuan, Jiangjie Chen, Changzhi Sun, Jiaqing Liang, Yanghua Xiao, and Deqing Yang. 2024. Analogykb: Unlocking analogical reasoning of language models with a million-scale knowledge base. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. 2024a. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b. *CoRR*.

Di Zhang, Jianbo Wu, Jingdi Lei, Tong Che, Jiatong Li, Tong Xie, Xiaoshui Huang, Shufei Zhang, Marco Pavone, Yuqiang Li, and 1 others. 2024b. Llama-berry: Pairwise optimization for o1-like olympiad-level mathematical reasoning. *arXiv preprint arXiv:2410.02884*.

A Appendix: Additional Details of Experiments

A.1 LLM Introductions

- Deepseek-V3 (DeepSeek-AI et al., 2025b): DeepSeek-V3 is a cutting-edge 671-billion-parameter open-source large language model (LLM) developed by DeepSeek, a Chinese AI company founded in 2023 by High-Flyer Capital Management, a quantitative hedge fund. It adopts a Mixture-of-Experts (MoE) architecture, activating only 37 billion parameters per token for efficient inference. Trained on 14.8 trillion tokens with a remarkably low cost of $5.576 million, it rivals top closed-source models like GPT-4o and Claude 3.5 Sonnet in performance, particularly excelling in math, coding, and Chinese tasks. The model is fully open-weight and available on Hugging Face and GitHub.

- QWQ-32B (QwenLM Team, 2023): QWQ-32B is a 32-billion-parameter inference model developed by the Alibaba Cloud's Tongyi Qianwen team, boasting performance comparable to DeepSeek-R1, which has 67.1 billion parameters. It leverages innovative techniques like RoPE embedding, SwiGLU activation functions, and RMSNorm layers, coupled with large-scale reinforcement learning, particularly excelling in math and coding tasks. QWQ32B is designed for efficient inference and can run on consumer-grade hardware like the Apple M4 Max chip, significantly lowering deployment barriers. It is open-sourced under the Apache 2.0 license and available on platforms like Hugging Face and ModelScope, making it accessible for both commercial and research purposes with comprehensive fine-tuning and deployment documentation. This model represents a significant breakthrough, offering high performance at a lower cost and wider accessibility.

- Deepseek-R1-Distilled-Qwen-32B (DeepSeek-AI et al., 2025a): DeepSeek-R1-Distilled-Qwen-32B is a cutting-edge 32-billion-parameter language model developed by DeepSeek AI through knowledge distillation from its larger 671-billion-parameter DeepSeek-R1 model. It leverages advanced techniques like reinforcement learning and optimized training strategies to achieve remarkable performance in tasks such as natural language understanding, code generation, and multilingual support. Designed for efficiency, this model significantly reduces computational and storage requirements while maintaining high accuracy, making it suitable for deployment on consumer-grade hardware. It is open-sourced under the Apache 2.0 license and available on platforms like Hugging Face and GitHub, offering a cost-effective solution for both research and commercial applications. This model represents a significant advancement in balancing performance and resource efficiency, enabling broader accessibility and practical deployment in various industries.

- GLM-Z1-32B (THUDM-GLM, 2025): GLM-Z1-32B is a cutting-edge 32-billion-parameter language model developed by Zhipu AI, based on its GLM-4-32B model. It incorporates advanced techniques such as cold-start strategies, extended reinforcement learning, and a unique "rumination" mechanism to enhance its reasoning capabilities. This model excels in tasks involving mathematics, coding, and logical reasoning, achieving remarkable performance with a reasoning speed of up to 200 tokens per second. Designed for efficiency, GLM-Z1-32B is optimized for lightweight deployment and commercial use, making it accessible for various applications. It is open-sourced and available on platforms like Hugging Face and ModelScope, offering a cost-effective solution for both research and practical deployment.

A.2 Datasets Details

Dataset. Our experimental evaluation utilized the following 4 benchmark datasets:

- ListFunction (Rule, 2020): This dataset is intended to assess the concept learning capabilities of both humans and machines by requiring the identification of a function that maps each input list to its corresponding output list.

- ARC2024 (Kaggle, 2024): The Abstraction and Reasoning Corpus (ARC), introduced by Chollet (Chollet, 2019), is a benchmark dataset designed to evaluate the generalizable reasoning capabilities of models. The 2024 Kaggle competition version of ARC (Kaggle, 2024) provides 400 training and 400 evaluation tasks. Since our experiments do not involve model training, we exclusively use the evaluation tasks,

which we refer to as ARC2024 throughout this paper.

- MiniARC (Kim et al., 2022): It is a compact dataset derived from the Abstraction and Reasoning Corpus (ARC), designed to measure abductive reasoning capabilities in AI models. It comprises 150 visual reasoning tasks with 5x5 input and output grids, focusing on diverse categories such as movement, color, object, number, geometry, and common sense. Mini-ARC maintains the complexity of the original ARC dataset while offering enhanced usability for model training due to its fixed size and clear problem-solving principles.
- ConceptARC (Moskvichev et al., 2023): A benchmark dataset designed to systematically evaluate the ability of both humans and AI systems to form and abstract concepts within the context of the Abstraction and Reasoning Corpus (ARC). ConceptARC consists of 16 "concept groups," each containing 10 tasks that instantiate a specific core concept (e.g., sameness, counting, movement) in various ways. This structure allows for a more nuanced assessment of generalization abilities compared to the original ARC dataset. The tasks involve transforming visual grids according to abstract rules, requiring solvers to infer the underlying concept from a few examples and apply it to novel situations.

Our experiment randomly selecte 110 samples from each of the 5 benchmarks. The total number of samples per benchmark shows in Table 2

Task	Total Sample Number
ListFunction	250
1D-ARC	910
ARC2024	400
MiniARC	200
ConceptARC	160

Table 2: Total sample number of different tasks.

A.3 Baseline Methods Details

We evaluate four baseline methods for comparison with MATSIR:

- Simple Transduction: This method only uses the simplest prompt to directly request a response from the large language model, and the inference result is obtained.
- Self-Consistency (SC) (Wang et al., 2023): A decoding strategy for chain-of-thought prompting in language models. It samples diverse reasoning paths and selects the most consistent answer, improving accuracy for complex reasoning tasks.
- Self-Refine (SR) (Madaan et al., 2023): The model generates an initial output, subsequently provides feedback on its own output, and refines the output based on this feedback. This iterative process continues until a predetermined stop condition is satisfied.
- Hypothesis Refine (HR) (Qiu et al., 2024a): This method involves two key strategies to enhance language model performance: 1) generating a large number of hypotheses and 2) employing iterative refinement with external feedback. The effectiveness of this method is improved by the combination of both iterative refinement and external feedback.

A.4 Hyperparameters of Main Experiments

The hyperparameters for MATSIR were set as follows,$a = 0.5$, $b = 0.5$, $c = 1.0$, with the maximum number of children-nodes set to 6, the maximum tree depth to 5, and the maximum number of iterations to 5. Additionally, the maximum retry attempts per agent were set to 3. And the maximum number of generated tokens was set to 8000. Random seed was set to 42.

A.5 Efficiency Comparison

As shown in Table 3, MATSIR consumes more computational resources than SC and HR, but is more efficient than SR. Moreover, the computational cost of each method varies across different tasks and backbone models. For instance, 1) in the ListFunction task using deepseek-v3 as the backbone, the number of generated characters by MATSIR is approximately twice that of HR. However, when using QWQ-32B as the backbone, the characters counts of MATSIR and HR are roughly the same; 2) While MATSIR and HR yield comparable character counts for the ListFunction task with QWQ-32B, MATSIR generates about four times as many characters as HR on the 1D-ARC task. These observations indicate that it is difficult to establish a consistent computational cost metric for fair comparisons across methods and models. On the other hand, existing works such as SC, SR, and HR also do not adopt a fixed computational bud-

Model	Method	ListFunction	1D-ARC	ARC2024	MiniARC	ConceptARC	AVG
Deepseek-V3	SC	11497.6	3400.8	2918.4	1154.3	1438.3	4081.9
	SR	80282.2	180802.6	156264.9	74369.9	109429.9	120229.9
	HR	35935.4	83962.8	67301.4	51184.4	61660.5	60008.9
	MARSIR	72817.4	81477.9	129401.9	115693.9	93923.4	98662.9
QWQ-32B	SC	8111.5	1756.2	1653.1	1154.3	1438.3	2822.7
	SR	156204.4	91913.4	156264.9	74369.9	109429.9	117636.5
	HR	71657.5	41003.5	67301.4	51184.4	61660.5	58561.4
	MATSIR	77355.0	158416.6	57627.4	125181.2	105275.8	104771.2
R1-Distill-Qwen-32B	SC	8111.5	1756.2	1653.1	1154.3	1438.3	2822.7
	SR	156204.4	91913.4	156264.9	74369.9	109429.9	117636.5
	HR	71657.5	41003.5	67301.4	51184.4	61660.5	58561.4
	MATSIR	55490.2	85553.2	65545.1	89413.5	79106.7	75021.7
GLM-Z1-32B	SC	11497.6	1756.2	1653.1	1154.3	1438.3	3499.9
	SR	80282.2	91913.4	156264.9	74369.9	109429.9	102452.0
	HR	35935.4	41003.5	67301.4	51184.4	61660.5	51417.0
	MATSIR	81281.6	140221.1	59588.9	119196.0	98250.3	99707.6

Table 3: Average output sequence lengths (in characters) for various methods, using different models as backbones, across multiple tasks. MATSIR (ours) are highlighted in gray, and the highest results in each column are underlined.

get for method comparison, likely because the underlying task itself remains an open and unsolved challenge.

A.6 Further Exploration of Rewarding Strategy

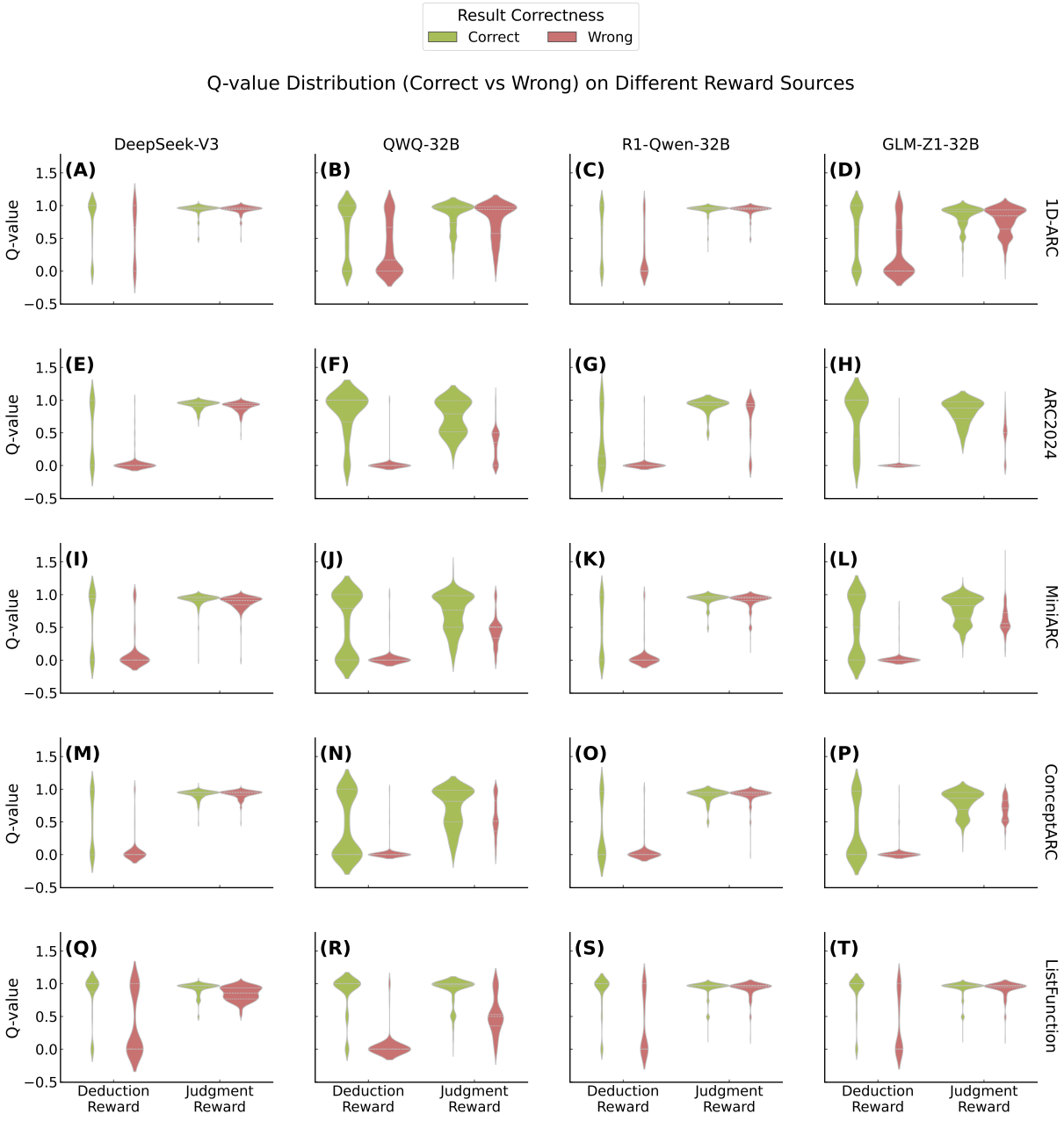

Figure 7: This figure shows the reward distribution of 4 models on 5 tasks.

A.7 Case Study

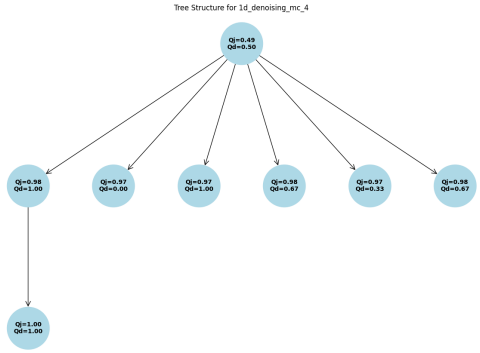

Figure 8: An example of successful early stopping.

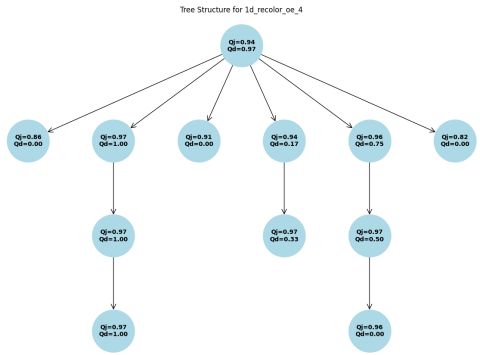

Figure 9: An example of finding a better refine path.

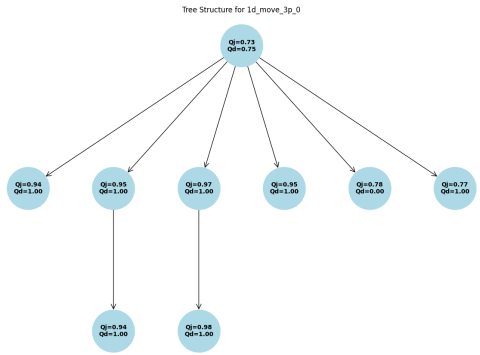

Figure 10: An example of finding a better refine path and successful early stopping.

A.8 Hardware Information

In our experiments, with the exception of DeepSeek-V3 which utilized a commercial API provided by a third-party company, the other three models (QwQ-32B, DeepSeek-R1-Distilled-Qwen-32B, and GLM-Z1-32B) were locally deployed and inferred using the LLaMA-Factory framework on our server. The hardware configuration of the server was as follows.

System Information

= System Information =
Operating System: Linux 6.5.0-28-generic
Kernel Version: #29~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Thu Apr 4 14:39:20 UTC 2
Hostname: klclab-X660-G45
Architecture: x86_64

= CPU Information =
CPU Model: x86_64
Physical Cores: 64
Logical Cores: 128
Max Frequency: 3.40 GHz

= Memory Information =
Total Memory: 1007.50 GB

= GPU Information =
Detected 8 GPU(s):
[GPU 0–7] NVIDIA A800-SXM4-80GB (79.33 GB, CUDA Capability 8.0, 108 Multiprocessors)

= NVIDIA Driver and CUDA Version =
CUDA Version: 12.4
CUDNN Version: 90100

= Software Environment =
Python Version: 3.10.14
PyTorch Version: 2.5.1+cu124