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

**Xiang Li**[1,2] **Yucheng Zhou**[3] **Xiangzhi Wei**[2] **Zesheng Shi**[1]
**Haiyuan Wan**[2,4] **Yifan Gong**[2] **Fangming Liu**[2]✉ **Jing Li**[1]✉
[1]Harbin Institute of Technology (Shenzhen) [2]Pengcheng Laboratory
[3]SKL-IOTSC, CIS, University of Macau [4]Tsinghua University
lixiang_alex@stu.hit.edu.cn,
fangminghk@gmail.com, jingli.phd@hotmail.com

## Abstract

Existing methods for enhancing the inductive reasoning of large language models (LLMs) at test-time typically depend on iterative self-refinement of hypotheses, which lacks explicit optimization guidance and effective error correction. This often results in superficial rewording and the accumulation of errors. To overcome these limitations, we propose MATSIR, a plug-and-play test-time framework integrating **M**ulti-**A**gent coordination with Monte Carlo **T**ree **S**earch to improve **I**nductive **R**easoning. MATSIR incorporates a dual-reward mechanism that provides explicit refinement signals, promoting logically coherent and semantically enriched hypotheses rather than mere rephrasing. Furthermore, it enables trajectory-level error correction through backtracking and pruning, allowing the system to recover from erroneous intermediate hypotheses. Experiments on five benchmarks across four LLMs show that MATSIR consistently outperforms previous best methods, yielding the highest average improvement of +4.9% on QWQ-32B and all-round improvement on Deepseek-V3. Our findings highlight that explicit guided search with built-in error correction is essential for advancing inductive reasoning in LLMs. Code is available at https://github.com/SolarWindRider/MATSIR.

## 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 method (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). Recently, as the capabilities of large language models have been continuously strengthened (Wei et al., 2022a; Qin et al., 2024), researchers have begun to investigate whether large language models can perform logical reasoning comparable to 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; Cai et al., 2025). One important criterion for evaluation is inductive reasoning, as it plays a crucial role in assessing a models ability to infer underlying patterns and generalize beyond observed data.

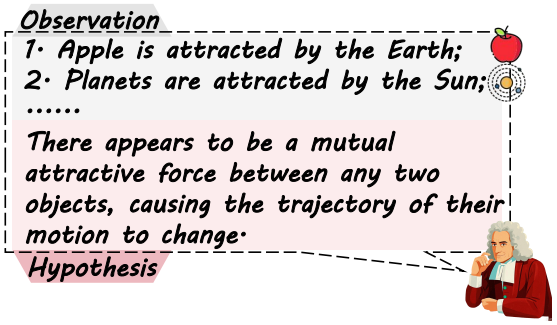


Figure 1: A concise example of human induction.

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., 2024; Qiu et al., 2024a; 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

✉ Corresponding authors.



*Findings of the Association for Computational Linguistics: ACL 2026*, pages 23548–23562
July 2-7, 2026 

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 underlying 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., 2024a).

Existing methods mainly focus hypothesis refinement on test-time. By leveraging multi-round reasoning and external verification, these approaches attempt to compensate for the limitations of large language models in inductive reasoning consistency and generalization. However, current methods suffer from several critical issues:

(1) Lack of reliable guidance in hypothesis refinement. Most existing studies allow large language models to refine hypotheses solely based on outputs from previous rounds, without providing explicit or informative signals to guide the refinement direction. As a result, the refinement process often degenerates into superficial rephrasing rather than genuine hypothesis optimization.

(2) Absence of error-correction mechanisms during refinement. Without an effective correction mechanism, existing approaches are prone to error accumulation, leading models to expend substantial computation on incorrect reasoning trajectories while failing to reach correct conclusions.

(3) Limited diversity of evaluated models. State-of-the-art methods are typically evaluated only on GPT-4, without validating their effectiveness across a broader range of model architectures which limits the generality of their conclusions.

To enhance the inductive reasoning capabilities of large language models, we propose 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 plug-and-play framework that integrates multi-agent coordination with Monte Carlo Tree Search (MCTS) to enhance inductive reasoning in large language models without additional fine-tuning.
- To overcome the lack of reliable guidance in hypothesis refinement, we introduce a **multi-agent dual-reward mechanism** that provides explicit optimization signals during refinement. This design encourages agents to generate genuinely improved hypotheses (rather than superficial paraphrases) by jointly enforcing logical coherence and semantic faithfulness.
- To mitigate error accumulation during iterative refinement, we incorporate **Monte Carlo Tree Search** as the underlying search strategy. It enables trajectory-level correction, allowing the framework to recover from early mistakes.
- Extensive experiments and analyses demonstrate that MATSIR not only effectively prunes the hypothesis space but also automatically discovers more reliable refinement trajectories. Moreover, it generalizes across diverse LLM architectures and model scales, addressing the limited model diversity in prior evaluations.

## 2 Related Work

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. (2024a) 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. (2024) 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. Complementary to these efforts, (Cai et al., 2025) has critically examined the role of chain-of-thought prompting in inductive reasoning, showing that ex-

plicit multi-step reasoning can even degrade performance on rule induction tasks due to error amplification across intermediate reasoning stages, which highlights the necessity of error correction mechanisms during the reasoning process.

## 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 Validation Sample**: The last one of the Rearranged demonstrations is reserved as a validation sample. This allows **Deductor** to employ deductive reasoning to verify the 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 **Deductor** 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, 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

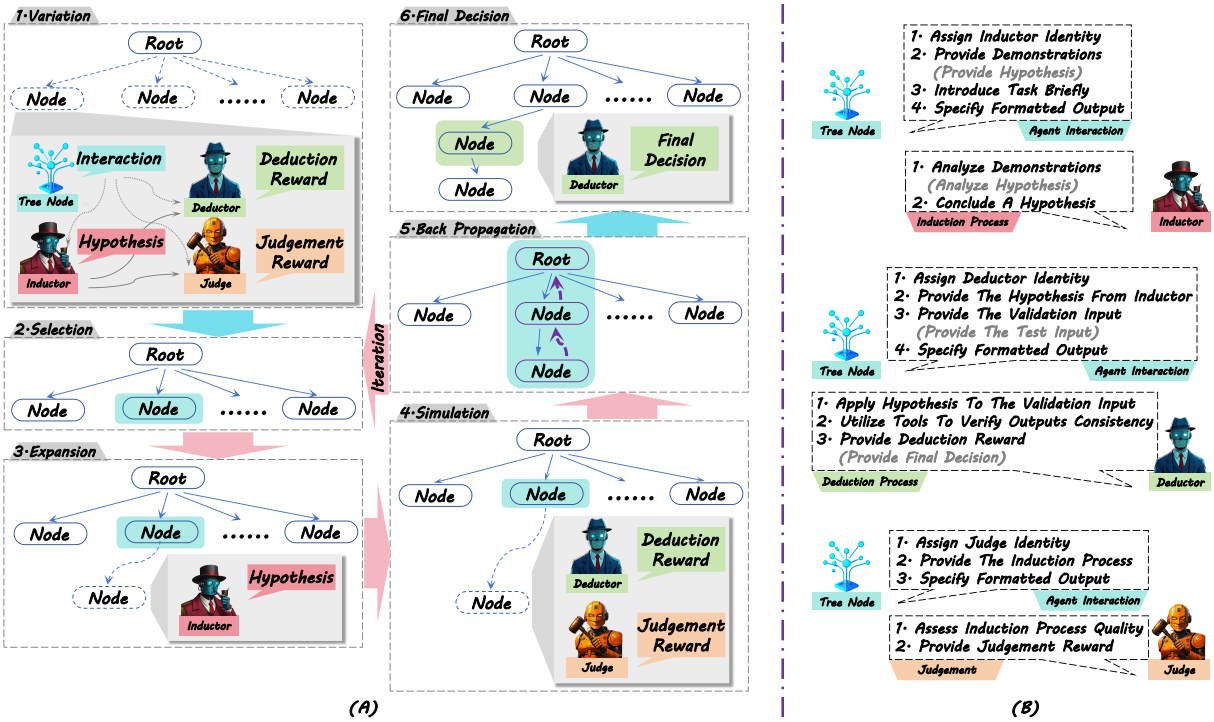


Figure 2: Overview of the MATSIR framework. (A) shows the customized Monte Carlo tree search process for inductive reasoning, where each node represents a hypothesis derived from rearranged demonstrations. The search explores and refines hypotheses through Variation, Selection, Expansion, Simulation, 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 Judgment 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_hypothesis(`$n_{root}$`)`**:**
  $n_h \leftarrow$ **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 ground truth of the verification sample, computing the Deduction Reward (Equation 4), while **Judge** assesses inductive 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 recursively propagates values up to the root, enabling each nodes Q-values to reflect the best observed outcomes.

**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

**Models.** 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).

**Datasets.** For each benchmark, we randomly sample 110 instances for evaluation:

- 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 in this paper.

- 1D-ARC (Xu et al., 2024b): A simplified variant of ARC, designed to study abstract reasoning capabilities of large language models in a more tractable setting. Unlike the original ARC, which operates on two-dimensional grids, 1D-ARC reduces each task to a one-dimensional sequence (i.e., a row of colored cells), while preserving core reasoning primitives such as pattern transformation, object manipulation, and rule induction.
- 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.

**Baseline Methods.** We evaluate five baseline methods for comparison with MATSIR:

- Simple Transduction: This method only uses the simplest prompt to directly request a response from a large language model to obtain inference result (Li et al., 2025).
- 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., 2024b): 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.
- Hypothesis Search (HS) (Wang et al., 2024): This method first proposes diverse natural-language hypotheses describing potential underlying rules, then translates them into executable programs. These programs are verified by execution against the training examples, and only those consistent with all observations are used for generalization. By decoupling abstract hypothesis formation from concrete program execution, the approach enables more reliable and interpretable inductive reasoning compared to direct prompting.

**Hyperparameters.** $a = 0.5$, $b = 0.5$, $c = 1.0$, 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. The random seed was set to 42.

### 4.2 Main Results

Table 1 reports the main experimental results of MATSIR compared with strong baselines across four LLMs and five inductive reasoning benchmarks. Overall, MATSIR consistently achieves the highest average performance across all evaluated backbones, demonstrating both robust effectiveness and strong cross-model generalization.

Across all four backbone models, MATSIR delivers clear and consistent improvements in average accuracy, with gains of +4.2, +4.9, +2.1, and +3.0 points over the strongest existing baselines for Deepseek-V3, QWQ-32B, R1-Qwen-32B, and GLM-Z1-32B, respectively. This trend indicates that MATSIR is not tailored to a specific architecture, but instead provides a generally applicable enhancement to inductive reasoning capability.

Notably, MATSIR outperforms prior reasoning-

| 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., 2024b) | 77.3 | 5.4 | 64.5 | 18.2 | 25.4 | 38.2 |
| | HS [ICLR2024] (Wang et al., 2024) | 86.4 | 27.3 | 80.9 | 30.0 | 24.5 | 49.8 |
| | **MATSIR** [OURS] | **87.3** | **30.9** | **89.1** | **31.8** | **30.9** | **54.0** |
| | *(Improvement)* | (↑ 0.9) | (↑ 3.6) | (↑ 8.2) | (↑ 1.8) | (↑ 5.5) | (↑ 4.2) |
| 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., 2024b) | 63.6 | 5.4 | 61.8 | 18.2 | 25.4 | 34.9 |
| | HS [ICLR2024] (Wang et al., 2024) | 83.6 | 5.5 | 57.3 | 23.6 | 20.0 | 38.0 |
| | **MATSIR** [OURS] | **91.5** | **5.5** | **61.8** | **39.6** | **16.2** | **42.9** |
| | *(Improvement)* | (↑ 7.9) | (↓ 3.6) | (↑ 0.0) | (↑ 12.3) | (↓ 9.2) | (↑ 4.9) |
| 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., 2024b) | 63.6 | 5.4 | 61.8 | 18.2 | 25.4 | 34.9 |
| | HS [ICLR2024] (Wang et al., 2024) | 81.8 | 4.5 | 50.0 | 20.0 | 17.3 | 34.7 |
| | **MATSIR** [OURS] | **86.6** | **4.5** | **57.3** | **23.6** | **13.0** | **37.0** |
| | *(Improvement)* | (↑ 4.8) | (↓ 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., 2024b) | 77.3 | 5.4 | 61.8 | 18.2 | 25.4 | 37.6 |
| | HS [ICLR2024] (Wang et al., 2024) | 80.9 | 4.5 | 54.5 | 22.7 | 19.1 | 36.3 |
| | **MATSIR** [OURS] | **79.5** | **6.4** | **62.7** | **38.2** | **16.4** | **40.6** |
| | *(Improvement)* | (↓ 1.4) | (↓ 2.7) | (↑ 0.9) | (↑ 15.5) | (↓ 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. *(Improvement)* denotes performance change of MATSIR compared to the current highest score, with (↑ *green*) representing improvement and (↓ *red*) indicating regression.

enhancement methods such as Hypothesis Refinement (HR), and Hypothesis Search (HS) on all models in terms of overall average performance, despite these baselines already incorporating iterative or hypothesis-driven reasoning mechanisms.

### 4.3 Empirical Analysis

**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 illustrates 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 number of nodes decreases progressively as the tree depth increases. This indicates that the search tree

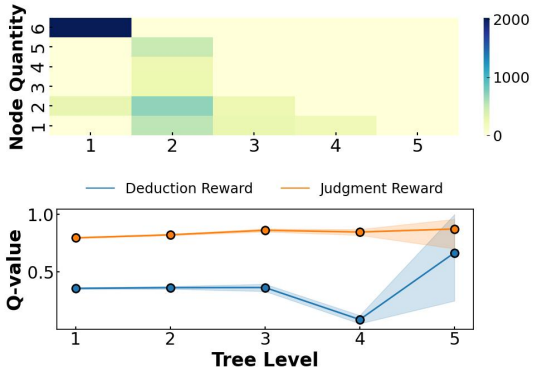


Figure 3: Node quantity (upper) and average values of 2 rewards (lower) as search trees deepen.

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

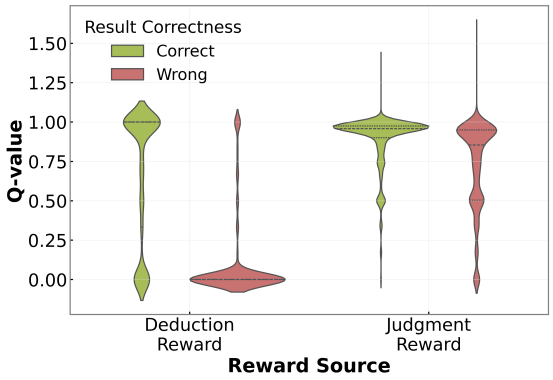


Figure 4: Distributive patterns of different rewards on best-hypotheses-nodes and their association with final results correctness

process progressively optimizes the hypotheses.

Figure 9 in Appendix A.5 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 to improve the 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 values, even in cases where the final result is wrong. Figure 7 in Appendix A.4 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).

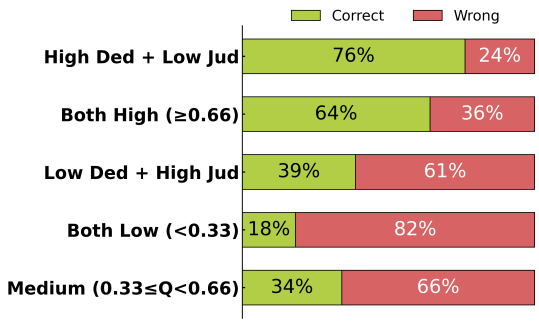


Figure 5: Proportion of correct and wrong responses across five conditions.

**Finding 2:** Compared with the LLM-as-Judge paradigm, verifiable rewards perform 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 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**

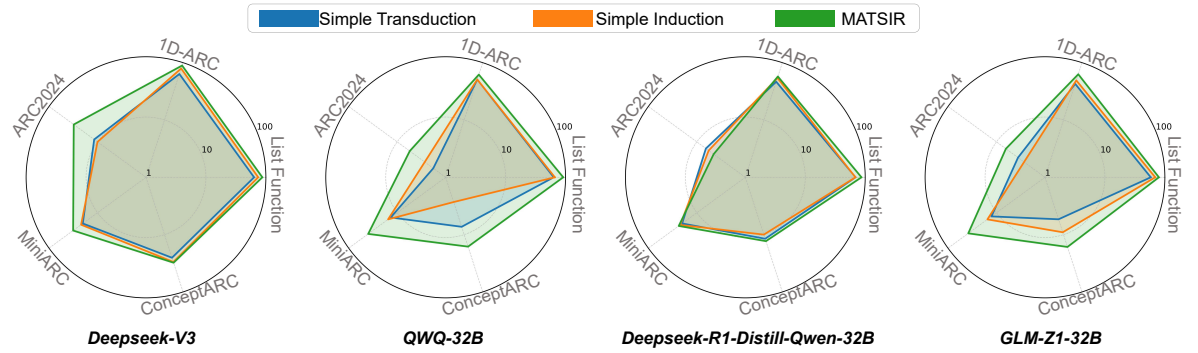


Figure 6: Comparative performance assessed on 5 different tasks, presented in **log-scale** radar plots.

**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

We present MATSIR, a plug-and-play framework that significantly advances inductive reasoning in LLMs through guided search and error correction. By integrating MCTS with multi-agent coordination, we address the critical "blind spots" of existing self-refinement methods, specifically the lack of optimization guidance and the inability to prune erroneous hypotheses. Our extensive experiments show consistent improvements across multiple benchmarks, most notably on QWQ-32B and DeepSeek-V3, proving that MATSIR is both model-agnostic and highly effective. As LLMs are increasingly deployed in complex reasoning scenarios, the principles of explicit search and trajectory-level recovery established by MATSIR provide a scalable pathway toward more robust and autonomous machine intelligence.

## Acknowledgements

This work was supported in part by the Major Key Project of PCL under Grant PCL2025A10 and PCL2024A06, in part by the Shenzhen Science and Technology Program under Grant RCJC20231211085918010, JCYJ20241202123503005, GXWD20231128103232001, ZDSYS20230626091203008, and KQTD20240729102154066, in part by the National Natural Science Foundation of China (62476070), and in part by the Department of Science and Technology of Guangdong (2024A1515011540).

## Limitation

While MATSIR effectively enhances inductive reasoning through explicit guided search, its current implementation has certain limitations. The integration of Monte Carlo Tree Search and multi-agent interactions results in increased inference costs, presenting a trade-off between state-of-the-art reasoning performance and computational efficiency. Additionally, the reliability of the "Judgment Reward" is somewhat sensitive to the base model's intrinsic reasoning capabilities, which may vary across different architectures or scales.

## References


Chengkun Cai, Xu Zhao, Haoliang Liu, Zhongyu Jiang, Tianfang Zhang, Zongkai Wu, Jenq-Neng Hwang, and Lei Li. 2025. The role of deductive and inductive reasoning in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

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, and Baobao Chang. 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.

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.

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.

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

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.

Wen-Ding Li, Keya Hu, Carter Larsen, Yuqing Wu, Simon Alford, Caleb Woo, Spencer M. Dunn, Hao Tang, Wei-Long Zheng, Yewen Pu, and Kevin Ellis. 2025. Combining induction and transduction for abstract reasoning. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

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

Suvir Mirchandani, Fei Xia, Pete Florence, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. 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)*.

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, and Nouha Dziri. 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 Xiang Ren. 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.

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.

Ruocheng Wang, Eric Zelikman, Gabriel Poesia, Yewen Pu, Nick Haber, and Noah Goodman. 2024. 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, and Denny Zhou. 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 (EACL)*.

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)*.

# 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

Our experiments 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 Efficiency Comparison

As shown in Table 3, MATSIR consumes more computational resources than SC and HR, but is

| 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.

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

### A.4 Further Exploration of Rewarding Strategy

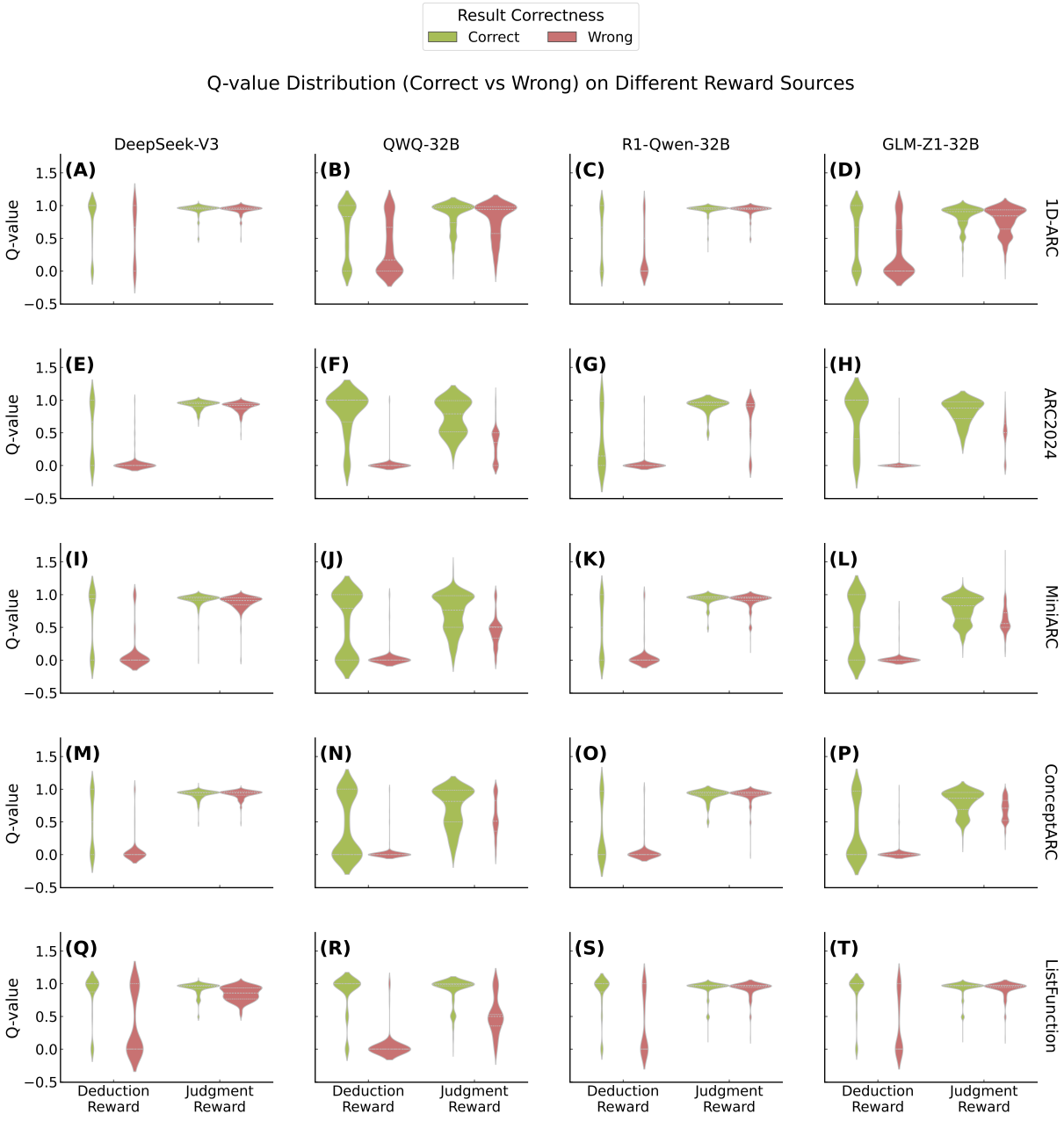


Figure 7: This figure shows the reward distribution of 4 models on 5 tasks.

## A.5 Case Study

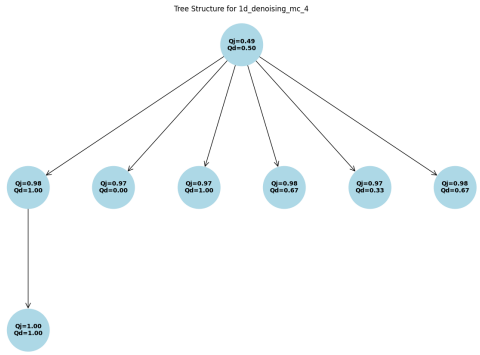


Figure 8: An example of successful early stopping.

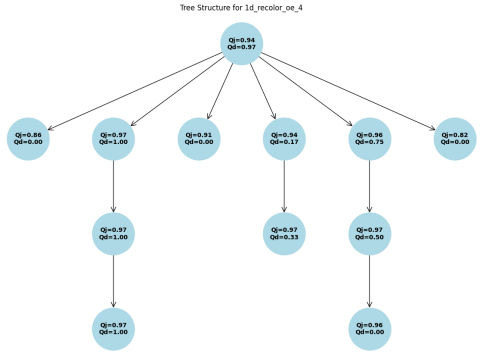


Figure 9: An example of finding a better refine path.

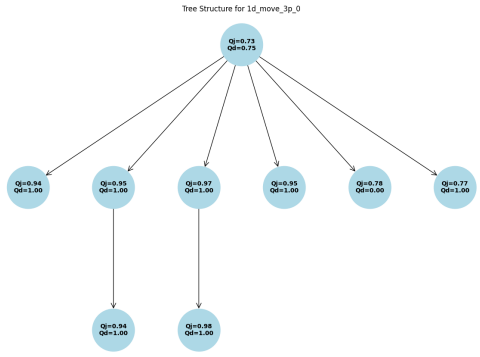


Figure 10: An example of finding a better refine path and successful early stopping.

## A.6 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