Mixture-of-Minds: Multi-Agent Reinforcement Learning for Table Understanding

Yuhang Zhou [1] **Mingrui Zhang** [1] **Ke Li** [1] **Mingyi Wang** [1] **Qiao Liu** [1]
Qifei Wang [1] **Jiayi Liu** [1] **Fei Liu** [1,2] **Serena Li** [1] **Weiwei Li** [1] **Mingze Gao** [1]
Abhishek Kumar [1] **Xiangjun Fan** [1] **Zhuokai Zhao** [1†] **Lizhu Zhang** [1†]
[1] Meta AI [2] Emory University
{zyhang, zhuokai, lizhu}@meta.com

Abstract

Understanding and reasoning over tables is a critical capability for many real-world applications. Large language models (LLMs) have shown promise on this task, but current approaches remain limited. Fine-tuning based methods strengthen language reasoning; yet they are prone to arithmetic errors and hallucination. In contrast, tool-based methods enable precise table manipulation but rely on rigid schemas and lack semantic understanding. These complementary drawbacks highlight the need for approaches that integrate robust reasoning with reliable table processing. In this work, we propose MIXTURE-OF-MINDS, a multi-agent framework that decomposes table reasoning into three specialized roles: planning, coding, and answering. This design enables each agent to focus on a specific aspect of the task while leveraging code execution for precise table manipulation. Building on this workflow, we introduce a self-improvement training framework that employs Monte Carlo Tree Search (MCTS) rollouts to generate pseudo-gold trajectories and optimize agents with reinforcement learning (RL). Extensive experiments show that MIXTURE-OF-MINDS delivers substantial gains, reaching 62.13% on TableBench and surpassing GPT-o3-mini. These results demonstrate the promise of combining structured multi-agent workflows with RL to advance table understanding. Our code and data are available at https://github.com/Tonyzhou98/mixture-of-minds

1 Introduction

Understanding and reasoning over tables is a fundamental capability for many real-world applications, including finance, healthcare, and knowledge management (Zhang et al., 2025c; Cheng et al., 2025; Fang et al., 2024). Compared with free text, tables present unique challenges: they are dense, structured, and often contain noisy or incomplete entries. Effective table understanding therefore requires models to retrieve relevant cells, perform reasoning steps such as arithmetic or comparison, and map the results into natural language answers (Wang et al., 2024; Yang et al., 2025b).

Large language models (LLMs) have recently become the backbone of state-of-the-art table understanding approaches (Sui et al., 2024; Tang et al., 2025). Prior work generally follows two directions. One seeks to enhance the intrinsic reasoning capability of LLMs through model-only training, using techniques such as supervised finetuning (SFT) or reinforcement learning (RL) (Wu et al., 2025c). The other direction leverages external tools, most commonly Python or SQL, to transform tables, filter noise, and extract useful information before the LLM generates the final answer (Sun et al., 2023; Shi et al., 2024; Gao et al., 2023). The first stream provides flexible language-based reasoning but is prone to arithmetic errors (Bertolazzi et al., 2025), structural confusion (Orgad et al., 2024), and hallucination (Kalai et al., 2025), while the second enables precise table manipulation but relies on rigid schema and lacks semantic depth or intent understanding (Rath, 2025).

To address these challenges, we propose **MIXTURE-OF-MINDS**, a multi-agent framework that integrates the advantages of both directions. Instead of relying on a single agent to solve the task end-to-end, MIXTURE-OF-MINDS decomposes table reasoning into three specialized roles: a *planning agent* that outlines reasoning steps, a *coding agent* that generates and executes code to transform the table, and an *answering agent* that derives the final answer from structured evidence. This decomposition not only makes the workflow more interpretable but also allows each LLM agent to focus on a well-defined part of the problem.

† Joint last author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2424–2439
July 2-7, 2026

However, training such a multi-agent system is significantly more challenging than training a single agent. The key difficulty is the lack of intermediate supervision: while final answers can be supervised, there are usually no gold-standard plans or code traces to guide the planning and coding agents. To address this, we introduce a self-improvement training framework that leverages Monte Carlo Tree Search (MCTS)-style rollouts (Browne et al., 2012) to automatically collect high-quality intermediate trajectories as pseudo-gold supervision. MCTS explores many alternative reasoning paths and keeps only those that lead to a correct final answer. By assuming that the intermediate steps along successful paths are likely to be valid, we turn the hard problem of supervising reasoning into a sequence of verifiable subtasks. This enables the use of RL methods, such as Group Relative Policy Optimization (GRPO) (Shao et al., 2024), with specifically designed reward functions that jointly capture plan quality, code execution validity, and final answer accuracy.

Our contributions and findings can be summarized as follows:

- ✤ We propose a novel agent workflow for table understanding that decomposes question solving into planning, coding, and answering, with specialized agents for each stage.
- ✤ We propose a self-improvement training framework that leverages MCTS-based data generation to yield verifiable intermediate supervision for multi-agent optimization.
- ✤ By integrating both the workflow and training paradigm, we introduce MIXTURE-OF-MINDS, which achieves substantial improvements over strong baselines. Extensive experiments show that MIXTURE-OF-MINDS attains 62.13% accuracy on TableBench (Wu et al., 2025b) with smaller LLMs and surpasses state-of-the-art models such as GPT-o3-mini (Jaech et al., 2024).

2 Related Work

2.1 Tabular Data Understanding

Table Question Answering (Table QA) has advanced alongside the development of increasingly complex evaluation benchmarks (Mueller et al., 2019; Jin et al., 2022; Wang et al., 2024). Early work focused on simple fact checking (Pasupat and Liang, 2015; Iyyer et al., 2017; Chen et al., 2019); later efforts incorporated tasks requiring complex mathematical reasoning (Chen et al., 2021; Katsis et al., 2021); and recent benchmarks extend to multimodal settings that integrate tables with other modalities (Talmor et al., 2021). This evolution has expanded table understanding tasks from single-cell extraction to broader reasoning and calculations over entire table contexts.

To meet the need for reasoning over entire tables, prior work has generally followed two directions: (1) enhancing the reasoning ability of LLMs through model-only inference, using table-specific RL or SFT methods without external tools (Yang et al., 2025b; Wu et al., 2025c; Sui et al., 2024; Lei et al., 2025); and (2) leveraging external tools such as Python or SQL to transform tables, filter noise, and extract useful information before feeding the processed data back into LLMs (Zhang et al., 2024c; Wang et al., 2024; Ye et al., 2023; Zhang et al., 2024a). Compared with these approaches, our work integrates the advantages of both streams. We propose a novel agent workflow that first decomposes table reasoning into planning, coding, and answering, and introduce an MCTS-based training framework that systematically improves the agents under this design.

2.2 Agentic Reinforcement Learning

Agentic Reinforcement Learning (Agentic RL) views LLMs as learnable policies within sequential decision-making loops, where RL equips them with agentic capabilities for long-horizon reasoning and interaction in dynamic environments (Zhang et al., 2025a). Using methods such as Proximal Policy Optimization (PPO) (Schulman et al., 2017) or GRPO (Shao et al., 2024) to fine-tune agent LLMs, researchers have enhanced their planning, reasoning, and tool-use abilities (Lu et al., 2025; Schick et al., 2023; Wu et al., 2025a; Guo et al., 2025). These advances have led to powerful agents applicable across domains ranging from search, coding to mathematics (Feng et al., 2025; Li et al., 2024; Jin et al., 2025). In the field of tabular data understanding, most prior work has trained either a single agent end to end or multiple agents jointed in a single-round RL setup (Liu et al., 2025; Chen et al., 2025). In contrast, we decompose table reasoning into specialized sub-agents and introduce an MCTS-based data generation and training pipeline that optimizes each agent through targeted RL, enabling more effective self-improvement.

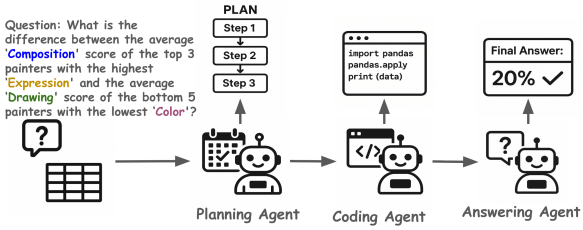


Figure 1: Overview of the MIXTURE-OF-MINDS workflow. The process is decomposed into three agents: (1) the *planning agent*, which outlines a structured plan; (2) the *coding agent*, which generates and executes code; and (3) the *answering agent*, which synthesizes the plan, table, and code outputs to produce the final answer.

3 MIXTURE-OF-MINDS Agents Workflow

Tables typically contain far more information than any single question requires, with many irrelevant entries that make it difficult for LLMs to locate useful facts and perform the necessary multi-hop reasoning. In such settings, relying on a single inference step to both plan, generate code, and produce the final answer often leads to incomplete reasoning, execution errors, or misaligned outputs.

To address these challenges, we propose MIXTURE-OF-MINDS, an agent workflow that leverages code as an external tool for precise table processing and decomposes the problem into three coordinated stages: *planning*, *coding*, and *answering*. As shown in Fig. 1, the planning agent first outlines a reasoning plan that guides the coding agent toward more accurate program generation. The coding agent then executes the program to transform the table into an intermediate representation, which serves as structured evidence. Finally, the answering agent combines this evidence with the question and the plan, relying on the model's reasoning and instruction-following abilities to produce the final answer. Each agent is guided by a specialized prompt, and together they form a modular and interpretable pipeline. Detailed prompt templates for each agent are provided in Appendix A.

Formally, let q denote a question and T its associated structured table. Let π_θ be the LLM-based agent parameterized by θ. Given a question–table pair (q, T), the workflow proceeds as follows:

Planning agent. The planning agent π_θ^{plan} generates a plan p wrapped in $\langle\texttt{plan}\rangle \cdot \langle/\texttt{plan}\rangle$ tags. The plan provides structured reasoning instructions that guide subsequent stages, helping prevent unguided and potentially ineffective exploration.

Coding agent. Conditioned on (q, T, p), the coding agent π_θ^{code} outputs executable code c, wrapped in Python markdown delimiters. Executing c yields an ordered sequence of intermediate table representations $\mathrm{T}' = (T'_1, T'_2, \ldots, T'_k)$, similar to (Wang et al., 2024). These intermediate results provide verifiable operations and reduce hallucinations common in free-form reasoning.

Answering agent. The answering agent π_θ^{ans} produces the answer $\hat{y}$, formatted within $\langle\texttt{answer}\rangle \cdot \langle/\texttt{answer}\rangle$ tags. By conditioning on (q, p, T'), the agent grounds its response in both the original query and the structured evidence.

This decomposition follows the natural workflow of table reasoning: planning the reasoning steps, deriving intermediate sub-results, and integrating them into a final answer. By explicitly separating these stages, the pipeline improves interpretability, allows error diagnosis at intermediate steps, and makes the reasoning process more robust to failures in any single component.

Improved test-time scaling support. Prior work has shown that allocating more test-time compute, such as sampling multiple trajectories or extending their length, can improve reliability and accuracy (Muennighoff et al., 2025; Zhang et al., 2025b). Unlike direct inference, which produces an answer in a single step, our workflow with task decomposition naturally increases token usage. This modular design and structure further makes our approach well suited to test-time scaling (TTS). Because each agent serves a distinct role, we can adjust decoding strategies, such as sampling more rollouts during early planning and coding to encourage diverse reasoning paths, and fewer rollouts during later answering to ensure stability. This fine-grained control enables early branching, a strategy known to enhance TTS (Yang and Holtzman, 2025; Fu et al., 2025), but difficult to realize in single-model setups, where one must repeat all chain-of-thought (CoT) traces or cannot easily determine which tokens require more exploration.

In the next section, we describe how we build a specifically-designed training framework to further enhance MIXTURE-OF-MINDS.

4 MIXTURE-OF-MINDS Agents Training

With the design of our MIXTURE-OF-MINDS agent workflow established, the next challenge is to iteratively improve the models within this pipeline.

We adopt a **step-by-step training strategy**, where the planning, coding, and answering agents are optimized sequentially and collaboratively. To provide reliable supervision at each stage, we employ MCTS–style rollouts (Browne et al., 2012) to generate diverse trajectories and extract high-quality intermediate plans and codes as gold-standard supervision (see Fig. 2). Each agent is then refined using GRPO (Shao et al., 2024) with structured and correctness-driven rewards. This progressive training ensures that agents improve in their designated roles while the overall workflow converges toward robust table reasoning.

4.1 MCTS-style Data Generation

To effectively finetune the planning, coding, and answering agents within the MIXTURE-OF-MINDS workflow, it is necessary to obtain gold-standard intermediate supervision, namely reference-quality plans and codes (we use the existing final answer as the label for the answering agent). Since such annotations are not available in existing datasets, we introduce a MCTS-style rollout procedure to automatically construct high-quality intermediate trajectories. Formally, following notations in §3, given a question-table pair (q, T), the process unfolds in three stages:

Plan generation. The planning agent π_θ^{plan} generates α candidate plans with temperature τ_p: $\{p_1, p_2, \ldots, p_\alpha\}$.

Code generation. For each plan p_i, the coding agent π_θ^{code} produces β executable code candidates with temperature τ_c: $\{c_{i,1}, c_{i,2}, \ldots, c_{i,\beta}\}$.

Answer generation. Each code candidate $c_{i,j}$ is executed to obtain intermediate table representations $\mathrm{T}'_{i,j}$. For every $(q, p_i, \mathrm{T}'_{i,j})$ tuple, the answering agent π_θ^{ans} generates γ candidate answers with temperature τ_a: $\{\hat{y}_{i,j,1}, \hat{y}_{i,j,2}, \ldots, \hat{y}_{i,j,\gamma}\}$.

A rollout trajectory is considered *successful* if at least one $\hat{y}_{i,j,k} \in \mathcal{A}(p_i, c_{i,j})$ matches the ground-truth answer $y^\star$. In such cases, both the intermediate plan p_i and code $c_{i,j}$ are considered as high-quality supervision signals. Therefore, the generated dataset $\mathcal{D}$ is defined as:

$$\mathcal{D} = \{(q, T, p_i, c_{i,j}, \hat{y}_{i,j,k}) \mid \exists (i,j,k) \text{ s.t. } \hat{y}_{i,j,k} = y^\star\}$$

which is used for self-improvement training of different agents. This procedure provides an automatic mechanism for collecting reliable intermediate annotations, thereby enabling stepwise optimization of the entire workflow.

4.2 Agent Training with GRPO

To optimize the agents in our workflow, we adopt GRPO (Shao et al., 2024), a RL algorithm that refines language model policies using relative reward signals within a group of responses. Specifically, GRPO refines a policy π_ϕ by sampling multiple candidate outputs for each input and updating the policy based on their relative quality. Formally, given a prompt x (e.g., $x = (q, T)$ for planning agent) from dataset $\mathcal{D}$, the old policy $\pi_{\phi_{\text{old}}}(y|x)$ samples G responses $\mathrm{Y}(x) = \{y_1, y_2, \ldots, y_G\}$, where each response $y_i \in \mathrm{Y}(x)$ is assigned a scalar reward $r(x, y_i)$ (defined differently for planning, coding, and answering agents as detailed in the later subsections). The group-normalized advantage is then computed as: $A(x, y_i) = \frac{r(x,y_i) - \bar{r}(x)}{\sigma_r(x)}$, where $\bar{r}(x)$ and $\sigma_r(x)$ are the mean and standard deviation of rewards within the group. The policy is then updated as:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q,\{o_i\}}\left[\frac{1}{G}\sum_{i=1}^{G}\min\left(\frac{\pi_\phi(y_i|x)}{\pi_{\phi_{\text{old}}}(y_i|x)}A_i,\right.\right.$$
$$\left.\left.\text{clip}\left(\frac{\pi_\phi(y_i|x)}{\pi_{\phi_{\text{old}}}(y_i|x)}, 1-\epsilon, 1+\epsilon\right)A_i\right)\right] - \beta\,\mathbb{D}_{\text{KL}}[\pi_\theta \| \pi_{\text{ref}}],$$

where ϵ is the clipping parameter and β controls the KL penalty against a reference policy π_{ref}.

In the context of our training framework, x denotes the input prompt for each agent as described in §3, while the candidate responses y_i correspond to either sampled plans, codes, or answers depending on the agent being optimized.

Planning agent training. For the planning agent, we first apply the MCTS-style rollout procedure (§4.1) with hyperparameters $(\alpha_{\text{plan}}, \beta_{\text{plan}}, \gamma_{\text{plan}})$ to generate trajectories for each (q, T). We retain only successful traces where the final answer matches $y^\star$, and use the extracted plans $\{p^\star\}$ as gold-standard supervision.

During GRPO training, the planning agent π_θ^{plan} generates G candidate plans in each group. Each plan p_i receives a reward $r^{\text{plan}}(p_i) = 0.1\, r^{\text{fmt}}(p_i) + 0.9\, r^{\text{BLEU}}(p_i, p^\star)$, where r^{fmt} checks correct formatting and r^{BLEU} is the BLEU score (Papineni et al., 2002) against the gold-standard plan.

Coding agent training. For coding, we leverage the fine-tuned planning agent to provide can-

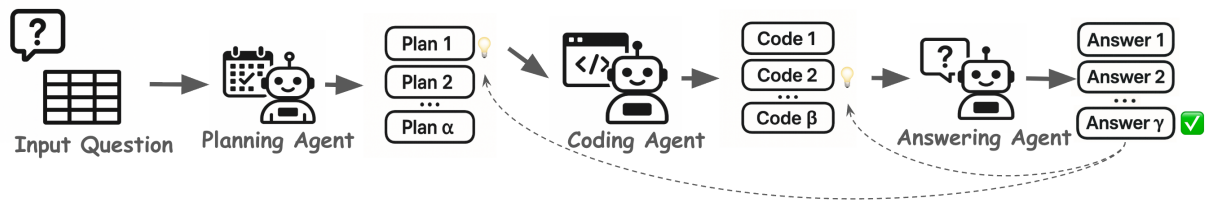


Figure 2: Illustration of the MIXTURE-OF-MINDS workflow with MCTS-style rollouts. Each agent produces multiple candidate inferences and we evaluate the final outputs to identify the correct answer. In this example, Answer γ is judged as the correct answer, and we trace back to Code 2 and Plan 1—the steps that produced it—which are then marked as pseudo-gold intermediate supervision for training.

didate plans, while the coding and answering agents remain base LLMs. MCTS rollouts with $(\alpha_{\text{code}}, \beta_{\text{code}}, \gamma_{\text{code}})$ yield trajectories, from which we keep only successful traces and extract gold-standard codes $\{c^\star\}$, using $(q, T, p^\star)$ as the corresponding training input.

In GRPO training, each candidate code c_i in the group with group size G is scored with a mixed reward: $r^{\text{code}}(c_i) = 0.1\, r^{\text{fmt}}(c_i) + 0.2\, r^{\text{exec}}(c_i) + 0.2\, r^{\text{op}}(c_i) + 0.5\, r^{\text{out}}(c_i, c^\star)$, where r^{fmt} checks Python markdown formatting, r^{exec} indicates successful execution, r^{op} measures code structural similarity to $c^\star$ (via Pandas-operation F1), and $r^{\text{out}}(c_i, c^\star)$ measures the alignment between the generated code output and the gold-standard code output, computed using BLEU score. We provide more details of the reward design in Appendix D.

Answering agent training. For the answering agent, both the planning and coding agents are fine-tuned, while the answering agent remains base. MCTS rollouts with $(\alpha_{\text{ans}}, \beta_{\text{ans}}, \gamma_{\text{ans}})$ generate trajectories, from which we keep correct answers $\{\hat{y}_i \mid \hat{y}_i = y^\star\}$ as supervision, using $(q, p^\star, \mathrm{T}'^\star)$ as the corresponding input prompt.

During GRPO training, each candidate answer $\hat{y}_i$ in the group with group size G is scored as $r^{\text{ans}}(\hat{y}_i) = 0.1\, r^{\text{fmt}}(\hat{y}_i) + 0.9\, r^{\text{EM}}(\hat{y}_i)$, where r^{fmt} checks special-tag formatting and r^{EM} denotes exact match with $y^\star$.

Together, these three stages form a step-wise training pipeline: the planning agent is first optimized to produce reliable plans, the coding agent is then trained using these plans as inputs, and finally the answering agent is refined with both plans and codes. This progressive strategy ensures that each agent benefits from the supervision generated by the previously fine-tuned components, leading to a consistent and robust end-to-end workflow.

5 Experiments

We design our experiments to rigorously evaluate the effectiveness of both the MIXTURE-OF-MINDS agents workflow and training framework.

5.1 Experimental Setup

Dataset. For training, we adopt TableInstruct (Wu et al., 2025b), which contains 4,897 unique question–answer pairs for table understanding. Details of the data pre-processing are provided in Appendix C. For in-domain evaluation, we use TableBench (Wu et al., 2025b), focusing on three representative datasets: Fact Checking, Numerical Reasoning, and Data Analysis, comprising a total of 836 test questions. To further assess generalization, we also evaluate on FinQA (Chen et al., 2022), which includes 1,147 financial contexts, questions, and tables, serving as an out-of-domain benchmark.

Baselines. We compare our framework against three categories of baselines: (1) Strong proprietary and open-source LLMs that are several orders of magnitude larger than ours, including GPT-4o (OpenAI, 2023), GPT-o3-mini (Jaech et al., 2024), Grok-4 (xAI, 2025), and DeepSeek-R1 (Guo et al., 2025); (2) Alternative training methodologies: we further consider models initialized from the same base LLMs but trained with different strategies, including GRPO and Decoupled Clip and Dynamic sampling Policy Optimization (DAPO) (Shao et al., 2024; Yu et al., 2025); and (3) state-of-the-art agent frameworks with integrated training recipes tailored for table understanding, such as Table-LLM (Zhang et al., 2024b) and Table-R1 (Wu et al., 2025c).

Models. We adopt a diverse set of LLMs with different architectures and sizes. Specifically, we use LLaMA-3.1-8B, LLaMA-3.3-70B (Grattafiori et al., 2024), Qwen-3-8B, Qwen-3-32B (Yang

Model	Setting	FC	NR	DA	Avg.
LLaMA-3.1-8B	Direct Inference	**54.17**	7.30	16.20	**15.42**
	MIXTURE-OF-MINDS	30.21	**8.08**	**19.77**	14.58
Qwen3-32B	Direct Inference	81.25	72.54	20.06	49.02
	MIXTURE-OF-MINDS	**81.25**	**73.99**	**20.06**	**49.68**
Qwen3-8B	Direct Inference	78.12	53.90	**24.11**	41.98
	MIXTURE-OF-MINDS	**79.17**	**68.18**	21.37	**47.38**
Gemma3-27B	Direct Inference	**79.17**	54.66	24.73	42.67
	MIXTURE-OF-MINDS	78.12	**61.11**	**30.93**	**47.83**
Nemotron-49B	Direct Inference	76.04	58.69	**33.48**	44.42
	MIXTURE-OF-MINDS	**76.04**	**71.28**	26.13	**50.30**
LLaMA-3.3-70B	Direct Inference	72.16	33.61	**34.53**	36.25
	MIXTURE-OF-MINDS	**78.12**	**69.52**	34.23	**52.88**

Table 1: Comparison between direct inference and MIXTURE-OF-MINDS agent workflow on Fact Checking (FC), Numerical Reasoning (NR), and Data Analysis (DA) tasks (Wu et al., 2025b).

et al., 2025a), Llama-3_3-Nemotron-Super-49B-v1_5 (Bercovich et al., 2025) and Gemma-3-27B (Team et al., 2025). This selection covers a wide range of LLM families (LLaMA, Qwen, and Nemotron) and sizes (8B-70B), providing a robust testbed for evaluating our agent workflow and training recipe.

Implementation details. The numbers of sampled plans (α), codes (β), and answers (γ) during MCTS rollouts differ by agent, as summarized in Table 5 in Appendix B. The temperature (τ_p, τ_c, τ_a) of MCTS and GRPO rollouts is fixed at 1.0. Other training details can be found in Appendix B.

5.2 Results

We begin by verifying that the MIXTURE-OF-MINDS agents workflow itself provides clear benefits over standard model-only inference. We then assess additional improvement gained through our MIXTURE-OF-MINDS training framework.

MIXTURE-OF-MINDS agent workflow significantly outperforms direct inference. We first verify the effectiveness of the MIXTURE-OF-MINDS pipeline compared to direct model-only inference on table understanding tasks. In this setting, the same base LLMs are evaluated under two conditions: (1) standard inference, where the model directly predicts the answer given the question and table with the CoT (Wei et al., 2022) prompts; and (2) the MIXTURE-OF-MINDS agents workflow, where the model operates through the planning, coding, and answering agents *without* training. This comparison isolates the impact of the workflow design itself, independent of our training framework.

From Table 1, we observe that the MIXTURE-OF-MINDS pipeline consistently improves performance over direct inference. For strong models such as LLaMA-3.3-70B and Nemotron-49B, the average improves by +16.6% and +5.9%, respectively, driven primarily by large gains in numerical reasoning. Qwen3-8B and Gemma3-27B also show meaningful improvements of +5.4% and +5.2%, confirming that even mid-sized models benefit from structured agent reasoning. While LLaMA-3.1-8B remains roughly unchanged overall, it still benefits in numerical reasoning and data analysis.

These results demonstrate the effectiveness of the MIXTURE-OF-MINDS workflow: decomposing table reasoning into planning, coding, and answering stages yields a substantial improvement compared to direct model predictions, with a particularly strong impact on numerical reasoning tasks where structured intermediate steps are essential.

The MIXTURE-OF-MINDS training recipe further boosts performance, surpassing LLMs several orders of magnitude larger. Next, we evaluate the effectiveness of MIXTURE-OF-MINDS training recipe. We report in-domain (Fact Checking, Numerical Reasoning and Data Analysis tasks (Wu et al., 2025b)) and out-of-domain (FinQA (Chen et al., 2022)) results comparing our methods and the baselines in Table 2 and Table 6, respectively.

For Table-LLM and Table-R1 in Table 2, we report results as provided by Wu et al. (2025c,b). From Table 2, we see that large proprietary models such as GPT-o3-mini and DeepSeek-R1 set strong baselines, but our framework helps open-source small LLMs closes the gap. Notably, after applying our framework to Qwen3-32B, it raises its weighted average to 59.9%, essentially matching GPT-o3-mini (59.9%), the strongest baseline among proprietary models. Moreover, with test-time scaling (TTS), Qwen3-32B further improves to 62.13%, surpassing GPT-o3-mini. For more details about TTS, including both parallel and sequential strategies, we will introduce it in §6.2.

Compared with GRPO and DAPO, our framework consistently provides additional gains. On Qwen3-8B, GRPO and DAPO reach 53.7% and 52.6%, respectively, while our method improves this to 57.4%. When compared to prior table-specific models, our framework also shows clear advantages. For Qwen3-8B, MIXTURE-OF-MINDS reaches 57.4%, surpassing Table-R1 (49.3%) and Table-LLM (35.9%) by a large margin. For

Model	Method	FC	NR	DA	Average
GPT-4o	Direct Inference	81.25	68.51	32.09	51.96
GPT-o3-mini	Direct Inference	86.46	82.07	35.56	59.90
Grok-4	Direct Inference	83.33	73.80	40.53	57.80
DeepSeek-R1	Direct Inference	82.29	75.51	35.06	56.31
Qwen3-8B	Direct GRPO	82.29	78.34	24.98	53.69
	Direct DAPO	81.25	75.31	25.84	52.58
	Table-R1	–	–	–	49.30
	Table-LLM	–	–	–	35.88
	MIXTURE-OF-MINDS	84.38	74.75	38.29	57.44
	MIXTURE-OF-MINDS (w/ TTS)	**86.46**	**77.83**	**41.59**	**60.35**
LLaMA-3.1-8B	Direct GRPO	81.25	48.87	18.49	37.86
	Direct DAPO	77.08	43.83	15.51	34.00
	Table-R1	–	–	–	42.78
	Table-LLM	–	–	–	30.77
	MIXTURE-OF-MINDS	70.83	59.70	31.69	46.72
	MIXTURE-OF-MINDS (w/ TTS)	**81.25**	**67.51**	**36.80**	**53.31**
Gemma3-27B	Direct GRPO	**84.38**	67.51	23.14	48.36
	Direct DAPO	80.21	65.74	21.69	46.57
	MIXTURE-OF-MINDS	79.17	68.26	34.46	52.50
	MIXTURE-OF-MINDS (w/ TTS)	83.33	**69.70**	**36.99**	**54.55**
Qwen3-32B	Direct GRPO	86.46	**80.10**	21.82	53.73
	Direct DAPO	86.46	77.08	26.18	54.08
	MIXTURE-OF-MINDS	85.42	76.57	42.12	59.90
	MIXTURE-OF-MINDS (w/ TTS)	**87.50**	78.34	**45.30**	**62.13**
LLaMA-3.3-70B	Direct GRPO	79.17	63.48	25.23	46.81
	Direct DAPO	75.83	63.94	22.61	45.65
	MIXTURE-OF-MINDS	82.29	70.53	36.97	55.04
	MIXTURE-OF-MINDS (w/ TTS)	**83.33**	**73.80**	**39.16**	**57.27**

Table 2: Comparison of our training framework against baselines on Fact Checking (FC), Numerical Reasoning (NR), and Data Analysis (DA) tasks. Table-R1 and Table-LLM results are directly extracted from Wu et al. (2025c) (CoT prompts). For single-pass inference, all models are evaluated with temperature 0, while parallel scaling is conducted with temperature 1. The largest values within each model block are bolded.

LLaMA-3.1-8B, our method boosts performance to 46.7%, making it far ahead of Table-R1 (42.8%) and Table-LLM (30.8%). Moreover, we observe similar improvements on the out-of-domain FinQA dataset, where, for example, Qwen3-32B improves from 46.4% with GRPO to 58.1% with our framework, confirming that the benefits extend beyond the in-domain setting.

Overall, MIXTURE-OF-MINDS consistently achieves the best results across different model sizes, outperforming GRPO, table-specific baselines, and cutting-edge closed-source models.

6 Ablation Studies and Analysis

6.1 Ablation on Sequential Training

In this section, we analyze how each stage of our sequential training recipe contributes to the final performance. As the training recipe is applied stage by stage, we evaluate performance after each stage to quantify the incremental gains from optimizing the planning, coding, and answering agents.

As shown in Fig. 3, we see that sequential training steadily improves performance by training each agent. On LLaMA-3.1-8B, fact-checking accuracy remains flat at 66.7% with MIXTURE-OF-MINDS alone, but increases to 67.7% after training the coding agent and further to 70.8% after training the answering agent; numerical reasoning also improves step by step, from 46.5% to 54.7% to 59.7%. A similar pattern holds for Qwen3-8B, where fact-checking grows from 77.1% to 81.3% to 84.4%. Notably, bigger models also benefit from this incremental training: on LLaMA-3.3-70B, data analysis improves sequentially from 34.2% to 35.4% to 36.9%, and on Qwen3-32B, data analysis rises from 20.1% to 34.3% and finally to 42.1%. In 8 out of 12 model–task combinations, sequentially training the agents leads to monotonic improvements.

Among all tasks, training the answering agent

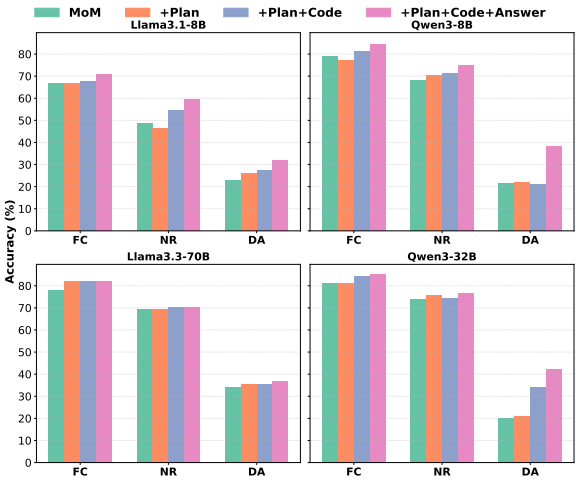


Figure 3: Ablation studies across different models. Each subplot shows performance on Fact Checking (FC), Numerical Reasoning (NR), and Data Analysis (DA) under sequential training stages: MIXTURE-OF-MINDS w/o training, +Plan agent training, +Plan+Code agent training, and +Plan+Code+Answer agent training.

Model	Setting	FC	NR	DA	Avg
Qwen3-8B	Base (Single)	78.12	53.90	24.11	41.98
	Base (Para.)	79.17	51.89	26.06	41.94
	MIXTURE-OF-MINDS (Single)	84.38	74.75	38.29	57.44
	MIXTURE-OF-MINDS (Seq.)	84.38	76.32	40.94	59.20
	MIXTURE-OF-MINDS (Para.)	**86.46**	**77.83**	**41.59**	**60.35**
Qwen3-32B	Base (Single)	81.25	72.54	20.06	49.02
	Base (Para.)	82.29	72.29	22.64	50.12
	MIXTURE-OF-MINDS (Single)	85.42	76.57	42.12	59.90
	MIXTURE-OF-MINDS (Seq.)	85.42	76.32	42.90	60.04
	MIXTURE-OF-MINDS (Para.)	**87.50**	**78.34**	**45.30**	**62.13**
LLaMA-3.1-8B	Base (Single)	54.17	7.30	16.20	15.42
	Base (Para.)	54.79	10.33	13.57	15.84
	MIXTURE-OF-MINDS (Single)	70.83	59.70	31.69	46.72
	MIXTURE-OF-MINDS (Seq.)	78.12	62.47	49.47	49.27
	MIXTURE-OF-MINDS (Para.)	**81.25**	**67.51**	**36.80**	**53.31**
Gemma3-27B	Base (Single)	79.17	54.66	24.73	42.67
	Base (Para.)	79.17	52.39	23.76	41.26
	MIXTURE-OF-MINDS (Single)	79.17	68.26	34.46	52.50
	MIXTURE-OF-MINDS (Seq.)	79.17	68.51	35.99	53.23
	MIXTURE-OF-MINDS (Para.)	**83.33**	**69.70**	**36.99**	**54.55**

Table 3: Test-time scaling analysis results on Fact Checking (FC), Numerical Reasoning (NR), and Data Analysis (DA) tasks. "Base" denotes direct model inference with base LLMs. We report performance under single-pass (1×), parallel scaling (8×), and sequential scaling (repeat until executable code).

tends to bring the most visible gains, especially in data analysis (e.g., Qwen3-8B improves from 21.2% to 38.3% and Qwen3-32B from 34.3% to 42.1%). This is expected because the answering agent directly aligns the final outputs with ground-truth answers, correcting residual errors even when planning and coding are accurate. In contrast, training the coding agent primarily improves execution validity and intermediate reasoning, yielding more moderate gains. This may also stem from the underlying coding capabilities of different LLMs, however, exploration in this direction is beyond the scope of this work. Taken together, these results demonstrate that while each agent contributes in different ways, the full sequence of planning, coding, and answering training delivers the largest overall improvement.

6.2 Test-time Scaling of MIXTURE-OF-MINDS

In this section, we explore how our proposed MIXTURE-OF-MINDS framework enables different forms of TTS. Specifically, we investigate two TTS strategies: (1) *parallel scaling*, where the workflow is executed multiple times and aggregated with self-consistency (Wang et al., 2022); and (2) *sequential scaling*, where the coding agent regenerates code until execution succeeds, adaptively increasing tokens per inference. This allows us to understand how MIXTURE-OF-MINDS trades additional inference-time compute for better performance.

For parallel scaling, we repeat the planning agent 8 times for early branching (temperature 1.0), while running the coding and answering agents once, resulting in 8 diverse traces that are aggregated by self-consistency (Wang et al., 2022). This is applied to both MIXTURE-OF-MINDS with our training framework and to direct inference as a baseline. For sequential scaling, we focus on MIXTURE-OF-MINDS with training: when the coding agent produces code that fails to execute, we regenerate by providing the historical code and error message until success. Results are reported in Table 3.

From Table 3, we find that applying TTS to MIXTURE-OF-MINDS consistently improves performance, boosting open-source models beyond GPT-o3-mini. Notably, Qwen3-32B with MIXTURE-OF-MINDS under parallel scaling achieves 62.1%, surpassing GPT-o3-mini's 59.9%, a substantial margin given the competitiveness of proprietary systems. On average, sequential scaling improves performance by +1.30%, and parallel scaling by +3.65%, with all models improving under both strategies. By contrast, model-only inference gains little from TTS, yielding only marginal gain and improving only half of the evaluated cases.

7 Conclusion

In this work, we introduced MIXTURE-OF-MINDS, a novel agent workflow for table reasoning, together with a compatible sequential training frame-

work that leverages MCTS-style data generation and GRPO optimization with designed reward functions. Through extensive experiments, we demonstrated that MIXTURE-OF-MINDS achieves substantial improvements over baseline methods, surpasses strong closed-source systems under test-time scaling, and consistently benefits from sequential training as shown in our ablation studies. These results highlight the value of structured, multi-agent reasoning. For future work, we expect that the proposed MCTS-based strategy for extracting intermediate supervision can extend to other domains that require complex agent collaboration.

8 Limitations

Although MIXTURE-OF-MINDS demonstrates strong performance, several limitations remain.

First, our study focuses exclusively on table understanding, and we do not experiment with multimodal settings such as charts, figures, or table–text integration (Talmor et al., 2021). Extending our framework to multimodal inputs would be an important step toward more general data reasoning.

Second, our coding agent is restricted to Python, which provides flexibility and rich libraries but may not be the most natural language for all table reasoning tasks. In many practical scenarios, SQL or domain-specific query languages are preferred, and adapting our workflow to support multiple coding backends is a promising future direction.

Finally, our training pipeline is built on GRPO, which is known to be sensitive to hyperparameter choices. More robust optimization techniques or hybrid objectives could further improve reliability.

We leave these extensions to multimodality, diverse tool usage, and more stable optimization as avenues for future work to broaden the applicability of MIXTURE-OF-MINDS.

9 Ethical Considerations

We do not identify any additional ethical concerns specific to this work beyond those generally associated with large language models. All experiments were conducted on publicly available datasets, and no sensitive or private data were used. We also note that AI assistants were employed to support coding tasks during the implementation of our experiments.

References

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, Ido Shahaf, Oren Tropp, Ehud Karpas, Ran Zilberstein, Jiaqi Zeng, Soumye Singhal, Alexander Bukharin, Yian Zhang, Tugrul Konuk, Gerald Shen, Ameya Sunil Mahabaleshwarkar, Bilal Kartal, Yoshi Suhara, Olivier Delalleau, Zijia Chen, Zhilin Wang, David Mosallanezhad, Adi Renduchintala, Haifeng Qian, Dima Rekesh, Fei Jia, Somshubra Majumdar, Vahid Noroozi, Wasi Uddin Ahmad, Sean Narenthiran, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Siddhartha Jain, Igor Gitman, Ivan Moshkov, Wei Du, Shubham Toshniwal, George Armstrong, Branislav Kisacanin, Matvei Novikov, Daria Gitman, Evelina Bakhturina, Jane Polak Scowcroft, John Kamalu, Dan Su, Kezhi Kong, Markus Kliegl, Rabeeh Karimi, Ying Lin, Sanjeev Satheesh, Jupinder Parmar, Pritam Gundecha, Brandon Norick, Joseph Jennings, Shrimai Prabhumoye, Syeda Nahida Akter, Mostofa Patwary, Abhinav Khattar, Deepak Narayanan, Roger Waleffe, Jimmy Zhang, Bor-Yiing Su, Guyue Huang, Terry Kong, Parth Chadha, Sahil Jain, Christine Harvey, Elad Segal, Jining Huang, Sergey Kashirsky, Robert McQueen, Izzy Putterman, George Lam, Arun Venkatesan, Sherry Wu, Vinh Nguyen, Manoj Kilaru, Andrew Wang, Anna Warno, Abhilash Somasamudramath, Sandip Bhaskar, Maka Dong, Nave Assaf, Shahar Mor, Omer Ullman Argov, Scot Junkin, Oleksandr Romanenko, Pedro Larroy, Monika Katariya, Marco Rovinelli, Viji Balas, Nicholas Edelman, Anahita Bhiwandiwalla, Muthu Subramaniam, Smita Ithape, Karthik Ramamoorthy, Yuting Wu, Suguna Varshini Velury, Omri Almog, Joyjit Daw, Denys Fridman, Erick Galinkin, Michael Evans, Katherine Luna, Leon Derczynski, Nikki Pope, Eileen Long, Seth Schneider, Guillermo Siman, Tomasz Grzegorzek, Pablo Ribalta, Monika Katariya, Joey Conway, Trisha Saar, Ann Guan, Krzysztof Pawelec, Shyamala Prayaga, Oleksii Kuchaiev, Boris Ginsburg, Oluwatobi Olabiyi, Kari Briski, Jonathan Cohen, Bryan Catanzaro, Jonah Alben, Yonatan Geifman, Eric Chung, and Chris Alexiuk. 2025. Llama-nemotron: Efficient reasoning models.

Leonardo Bertolazzi, Philipp Mondorf, Barbara Plank, and Raffaella Bernardi. 2025. The validation gap: A mechanistic analysis of how language models compute arithmetic but fail to validate it. *arXiv preprint arXiv:2502.11771*.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43.

Lei Chen, Xuanle Zhao, Zhixiong Zeng, Jing Huang, Liming Zheng, Yufeng Zhong, and Lin Ma. 2025. Breaking the sft plateau: Multimodal structured re-

inforcement learning for chart-to-code generation. *arXiv preprint arXiv:2508.13587*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2019. Tabfact: A large-scale dataset for table-based fact verification. *arXiv preprint arXiv:1909.02164*.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2022. Finqa: A dataset of numerical reasoning over financial data.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, et al. 2021. Finqa: A dataset of numerical reasoning over financial data. *arXiv preprint arXiv:2109.00122*.

Mingyue Cheng, Qingyang Mao, Qi Liu, Yitong Zhou, Yupeng Li, Jiahao Wang, Jiaying Lin, Jiawei Cao, and Enhong Chen. 2025. A survey on table mining with large language models: Challenges, advancements and prospects. *Authorea Preprints*.

Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun Qi, Scott Nickleach, Diego Socolinsky, Srinivasan Sengamedu, and Christos Faloutsos. 2024. Large language models (llms) on tabular data: Prediction, generation, and understanding–a survey. *arXiv preprint arXiv:2402.17944*.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. 2025. Deep think with confidence. *arXiv preprint arXiv:2508.15260*.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-sql empowered by large language models: A benchmark evaluation. *arXiv preprint arXiv:2308.15363*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

Nengzheng Jin, Joanna Siebert, Dongfang Li, and Qingcai Chen. 2022. A survey on table question answering: recent advances. In *China Conference on Knowledge Graph and Semantic Computing*, pages 174–186. Springer.

Adam Tauman Kalai, Ofir Nachum, Santosh S Vempala, and Edwin Zhang. 2025. Why language models hallucinate. *arXiv preprint arXiv:2509.04664*.

Yannis Katsis, Saneem Chemmengath, Vishwajeet Kumar, Samarth Bharadwaj, Mustafa Canim, Michael Glass, Alfio Gliozzo, Feifei Pan, Jaydeep Sen, Karthik Sankaranarayanan, et al. 2021. Ait-qa: Question answering dataset over complex tables in the airline industry. *arXiv preprint arXiv:2106.12944*.

Fangyu Lei, Jinxiang Meng, Yiming Huang, Tinghong Chen, Yun Zhang, Shizhu He, Jun Zhao, and Kang Liu. 2025. Reasoning-table: Exploring reinforcement learning for table reasoning. *arXiv preprint arXiv:2506.01710*.

Jia Li, Yunfei Zhao, Yongmin Li, Ge Li, and Zhi Jin. 2024. Acecoder: An effective prompting technique specialized in code generation. *ACM Transactions on Software Engineering and Methodology*, 33(8):1–26.

Shu Liu, Sumanth Hegde, Shiyi Cao, Alan Zhu, Dacheng Li, Tyler Griggs, Eric Tang, Akshay Malik, Kourosh Hakhamaneshi, Richard Liaw, Philipp Moritz, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. 2025. Skyrl-sql: Matching gpt-4o and o4-mini on text2sql with multi-turn rl.

Keer Lu, Chong Chen, Bin Cui, Huang Leng, and Wentao Zhang. 2025. Pilotrl: Training language model agents via global planning-guided progressive reinforcement learning. *arXiv preprint arXiv:2508.00344*.

Thomas Mueller, Francesco Piccinno, Massimo Nicosia, Peter Shaw, and Yasemin Altun. 2019. Answering conversational questions on structured data without logical forms. *arXiv preprint arXiv:1908.11787*.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.

OpenAI. 2023. Gpt-4 technical report.

Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. 2024. Llms know more than they show: On the intrinsic representation of llm hallucinations. *arXiv preprint arXiv:2410.02707*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*.

Amit Rath. 2025. Structured prompting and feedback-guided reasoning with llms for data interpretation. *arXiv preprint arXiv:2505.01636*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Liang Shi, Zhengju Tang, Nan Zhang, Xiaotong Zhang, and Zhi Yang. 2024. A survey on employing large language models for text-to-sql tasks. *ACM Computing Surveys*.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2024. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 645–654.

Ruoxi Sun, Sercan Ö Arik, Alex Muzio, Lesly Miculicich, Satya Gundabathula, Pengcheng Yin, Hanjun Dai, Hootan Nakhost, Rajarishi Sinha, Zifeng Wang, et al. 2023. Sql-palm: Improved large language model adaptation for text-to-sql (extended). *arXiv preprint arXiv:2306.00739*.

Alon Talmor, Ori Yoran, Amnon Catav, Dan Lahav, Yizhong Wang, Akari Asai, Gabriel Ilharco, Hannaneh Hajishirzi, and Jonathan Berant. 2021. Multimodalqa: Complex question answering over text, tables and images. *arXiv preprint arXiv:2104.06039*.

Zirui Tang, Weizheng Wang, Zihang Zhou, Yang Jiao, Bangrui Xu, Boyu Niu, Xuanhe Zhou, Guoliang Li, Yeye He, Wei Zhou, et al. 2025. Llm/agent-as-data-analyst: A survey. *arXiv preprint arXiv:2509.23988*.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

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

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, et al. 2024. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *arXiv preprint arXiv:2401.04398*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Mingyuan Wu, Jingcheng Yang, Jize Jiang, Meitang Li, Kaizhuo Yan, Hanchao Yu, Minjia Zhang, Chengxiang Zhai, and Klara Nahrstedt. 2025a. Vtool-r1: Vlms learn to think with images via reinforcement learning on multimodal tool use. *arXiv preprint arXiv:2505.19255*.

Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, Xeron Du, Di Liang, Daixin Shu, Xianfu Cheng, Tianzhen Sun, et al. 2025b. Tablebench: A comprehensive and complex benchmark for table question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25497–25506.

Zhenhe Wu, Jian Yang, Jiaheng Liu, Xianjie Wu, Changzai Pan, Jie Zhang, Yu Zhao, Shuangyong Song, Yongxiang Li, and Zhoujun Li. 2025c. Table-r1: Region-based reinforcement learning for table understanding. *arXiv preprint arXiv:2505.12415*.

xAI. 2025. Grok-4. Accessed: 2025-07-10.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Chenghao Yang and Ari Holtzman. 2025. How alignment shrinks the generative horizon. *arXiv preprint arXiv:2506.17871*.

Zheyuan Yang, Lyuhao Chen, Arman Cohan, and Yilun Zhao. 2025b. Table-r1: Inference-time scaling for table reasoning. *arXiv preprint arXiv:2505.23621*.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*, pages 174–184.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, Yifan Zhou, Yang Chen, Chen Zhang, Yutao Fan, Zihu Wang, Songtao Huang, Yue Liao, Hongru Wang, Mengyue Yang, Heng Ji, Michael Littman, Jun Wang, Shuicheng Yan, Philip Torr, and Lei Bai. 2025a. The landscape of agentic reinforcement learning for llms: A survey.

Han Zhang, Yuheng Ma, and Hanfang Yang. 2024a. Alter: Augmentation for large-table-based reasoning. *arXiv preprint arXiv:2407.03061*.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, et al. 2025b. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*.

Xiaokang Zhang, Sijia Luo, Bohan Zhang, Zeyao Ma, Jing Zhang, Yang Li, Guanlin Li, Zijun Yao, Kangli Xu, Jinchang Zhou, et al. 2024b. Tablellm: Enabling tabular data manipulation by llms in real office usage scenarios. *arXiv preprint arXiv:2403.19318*.

Xuanliang Zhang, Dingzirui Wang, Longxu Dou, Qingfu Zhu, and Wanxiang Che. 2025c. A survey of table reasoning with large language models. *Frontiers of Computer Science*, 19(9):199348.

Yunjia Zhang, Jordan Henkel, Avrilia Floratou, Joyce Cahoon, Shaleen Deep, and Jignesh M Patel. 2024c. Reactable: enhancing react for table question answering. *Proceedings of the VLDB Endowment*, 17(8):1981–1994.

Appendix

A Prompt Details

We present the prompt details for constructing our MIXTURE-OF-MINDS agent workflow in Table 4.

B Hyperparameter Details

The numbers of sampled plans (α), codes (β), and answers (γ) during MCTS rollouts differ by agent, as summarized in Table 5. For GRPO optimization, we use a group size of $G = 8$ (as defined in Section 4.2). The learning rate is set to 1×10^{-6}, with a global batch size of 256 and a rollout temperature of 1.0. Each experiment is trained for approximately 100 update steps.

Planning Agent (System)
You are an expert data analyst specializing in table analysis. Your task is to create a clear, step-by-step analysis plan to guide another LLM to generate pandas code to answer questions about tabular data. Guidelines: break down into logical steps, simple focused operations, actionable with pandas, no visualization, be specific, each step results in a new table closer to the answer, etc. Output enclosed in `<plan>...</plan>`.
Planning Agent (User)
Question and Input Table {instruction}. Your task is to create a detailed analysis plan with at most 4 steps. Each step should describe what to extract, what to do, and why it helps answer the question. Be specific about metrics, columns, or operations.
Coding Agent (System)
You are a coding agent using pandas. Write full Python code (with imports, table loading, and execution) to follow the given plan. No plotting. At the end, print (1) the answer and (2) a short summary. Output must be wrapped in "`python ...`".
Coding Agent (User)
Input table and question: {instruction}. The planner already made a plan: {plan}. Write code to execute it. Handle data types and empty cells carefully.
Answering Agent (System)
You are a data scientist specializing in table analysis. Answer questions with accuracy, critical thinking, and domain knowledge. Apply causal reasoning, avoid trivial correlations, be skeptical of artifacts, and follow requested answer format exactly.
Answering Agent (User)
Answer format, question, and table: {instruction}. **Plan**: {plan}. **Code output**: {code_output}. If errors occur in code output, omit it. Otherwise, verify correctness and provide the final answer.

Table 4: System and user prompt templates for planning, coding, and answering agents in MIXTURE-OF-MINDS.

Agent	α (Plans)	β (Codes)	γ (Answers)
Planning	8	4	1
Coding	1	8	1
Answering	1	4	8

Table 5: MCTS rollout hyperparameters for different agents, where α, β, and γ denote the number of sampled plans, codes, and answers, respectively.

C Data Processing

The TableInstruct (Shao et al., 2024) dataset was constructed from multiple existing table datasets,

including FinQA, TabFact, etc. Both questions and answers were constructed and validated using GPT-4-based agents. The answers were constructed with four approaches: DP (direct prompting), TCoT (textual chain-of-thought), SCoT (symbolic chain-of-thought) and PoT (program-of-thoughts), thus for each question, there are four answers in the dataset.

Since the questions/answers were only generated by LLM and not fully manually verified, we first investigated the dataset for correctness, and found two types of errors: 1) Answers generated with different approaches (DP / TCoT / SCoT / PoT) for the same question do not agree with each other. This appears mainly in Fact Checking and Numerical Reasoning questions, since there should only be one correct answer. 2) Wrong answers for Data Analysis categories. We identified several wrong answers in the dataset, such as mixing causality with correlation (e.g. the question is asking about causality, but the answer only provided correlation) and wrong value for trend forecasting.

Data cleaning for Fact Checking and Numerical Reasoning questions. We first applied rule-based filtering for Fact Checking and Numerical Reasoning data: for each question, the answer is correct, only the answers generated by all four methods are the same (DP / TCoT / SCoT / PoT). We normalize the format of the answers, such as numerical precision, before comparing them.

For the remaining questions, we used a coding agent to derive the answer. The coding agent produces runnable code to solve the problem and then answers the question from the code output. We used the Claude Sonnet 4 and Google Gemini 2.5 Pro models as the agent backend to generate two sets of answers for the same question.

We then compared the generated answers with the DP answer from the original dataset: if the DP answer matches one of the generated answers (either from Gemini or Claude agents output), we consider the DP answer correct. This resulted in 556 questions whose answers need to be updated. Next we check if the answer from two coding agents from Claude / Gemini agree with each other, and consider the generated answers as the correct one. The last 177 questions with different generated answers were reviewed by the authors and their final answers manually created.

Data cleaning for Data Analysis questions. Since the Data Analysis answers are provided in natural language, it is hard to use exact match to filter out wrong answers. We thus used LLM-as-judge design to extract the correct answers. We first applied a coding agent to generate a new answer for the question, then feeding both the original answer, the code output and the generated answer to another LLM to decide the final correct answer. We used Gemini Pro 2.5 as the LLM backend.

We then manually inspect a set of randomly sampled rows to ensure the data quality. This method is not 100% correct since the judge can make errors, yet we expect higher quality of the updated dataset, as of 1) the LLM base model has higher performance than the GPT-4 model used in the original paper published in 2024; 2) the coding agent providing a grounded reasoning process validated by external (python) tools. In the end, 1463 Data Analysis questions' answers were updated by the LLM.

D Coding Agent Reward Design Details

The coding agent reward consists of four components: format, execution, operation, and output rewards. For completeness, we provide further details here.

Format Reward (r^{fmt}). Ensures that generated code is wrapped within proper Python markdown, enabling consistent parsing and execution during evaluation.

Execution Reward (r^{exec}). Assigned as 1 if the generated code executes successfully without runtime errors. This enforces syntactic and runtime validity.

Operation Reward (r^{op}). We extract high-level Pandas operations (e.g., `groupby`, `merge`, `pivot`) from both the generated code c_i and gold code $c^\star$, normalize their order, and compute an F1 score over the operation set. This encourages structural similarity without requiring exact token-level matches.

Output Reward (r^{out}). Unlike r^{exec} (which checks only validity), this reward measures the *semantic correctness* of the generated code output relative to the gold-standard $c^\star$. We compute r^{out} as the BLEU score (Papineni et al., 2002) between the predicted output string and the reference output. This design allows partial credit for outputs that are semantically close but not exact matches,

and stabilizes training compared to a strict binary correctness reward.

Thus, the final coding agent reward balances surface validity (formatting, execution) with structural similarity and end-to-end correctness.

E Supplementary Results

E.1 FinQA Evaluation

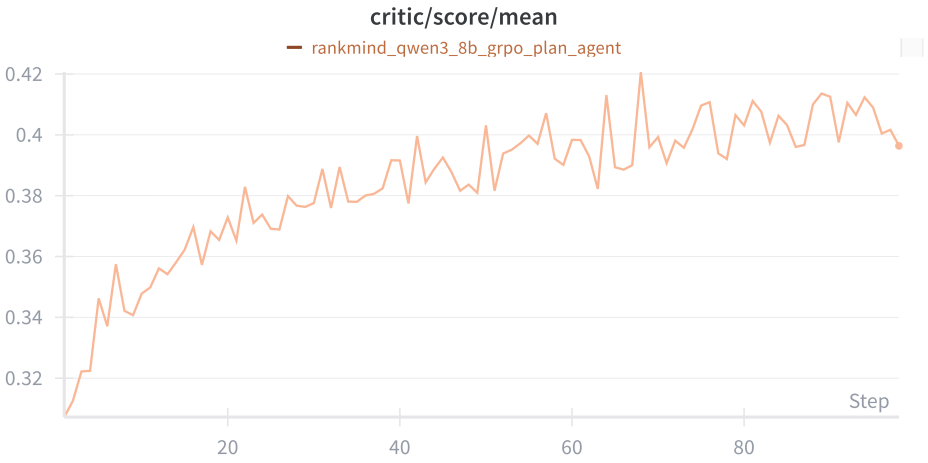
To better demonstrate the generalization ability of our training framework, we also evaluate the trained agents with MIXTURE-OF-MINDS on the FinQA dataset as an out-of-domain benchmark. The results are presented in Table 6.

E.2 Reward Curve Plot

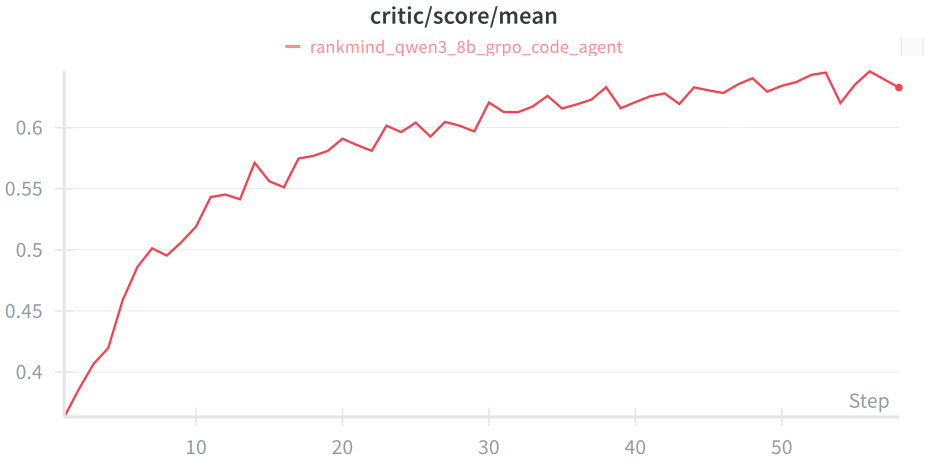
To better understand how each component contributes during training, we visualize the reward trajectories of the planning, coding, and answering agents on Qwen3-8B in Figure 4.

Model	Method	FinQA (%)
Qwen3-8B	Direct GRPO	44.57
	MIXTURE-OF-MINDS (w/o finetune)	54.63
	MIXTURE-OF-MINDS	**55.25**
LLaMA-3.1-8B	Direct GRPO	27.18
	MIXTURE-OF-MINDS (w/o finetune)	24.54
	MIXTURE-OF-MINDS	**33.80**
Gemma3-27B	Direct GRPO	50.93
	MIXTURE-OF-MINDS (w/o finetune)	48.46
	MIXTURE-OF-MINDS	**51.89**
Qwen3-32B	Direct GRPO	46.43
	MIXTURE-OF-MINDS (w/o finetune)	56.57
	MIXTURE-OF-MINDS	**58.08**
LLaMA-3.3-70B	Direct GRPO	50.57
	MIXTURE-OF-MINDS (w/o finetune)	48.10
	MIXTURE-OF-MINDS	**51.89**

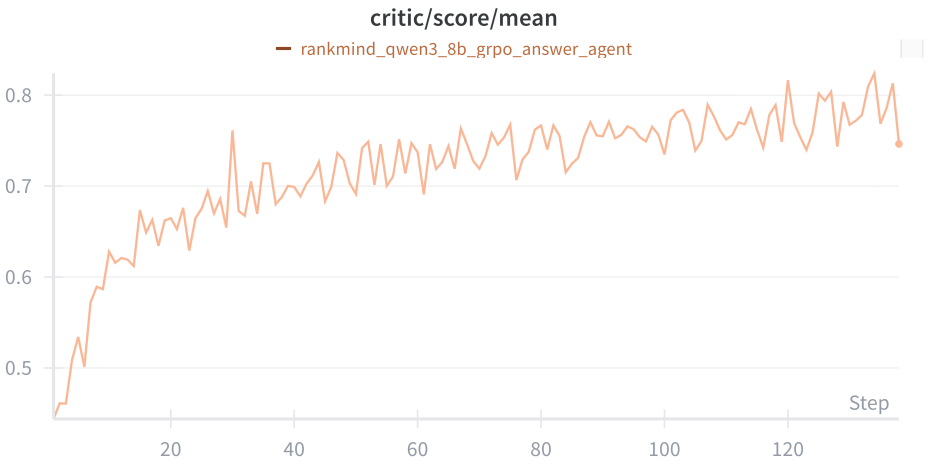
Table 6: Out-of-domain results on FinQA. We compare a direct GRPO baseline with our MIXTURE-OF-MINDS pipeline (without training) and MIXTURE-OF-MINDS with our training framework . Best result per model is bolded. All runs use a single inference pass with temperature 0.



(a) Planning Agent



(b) Coding Agent



(c) Answering Agent

Figure 4: Reward trajectories of the planning, coding, and answering agents on Qwen3-8B. Each plot illustrates how our designed reward functions guide learning over training.