TATTO: Tool-Augmented Thinking PRM for Tabular Reasoning

Jiaru Zou[1], Soumya Roy[2], Vinay Kumar Verma[2], Ziyi Wang[3], David Wipf[2], Pan Lu[4], Jingrui He[1], Sumit Negi[2]
[1]University of Illinois Urbana-Champaign, [2]Amazon,
[3]Purdue University, [4]Stanford University

Abstract

Test-time scaling has emerged as a promising paradigm to enhance reasoning in large reasoning models by allocating additional inference-time compute. However, its potential for tabular reasoning remains underexplored. We identify that existing process reward models, widely used to supervise reasoning steps, struggle with table-specific operations such as table retrieval and schema interaction, leading to bottlenecked performance under TTS. To address this gap, we propose TATTO, the first table-grounded PRM framework that leverages tool use for accurate verification. We develop a scalable data curation pipeline producing over 60k high-quality step-level annotations that combine expert rationales with programmatic tool executions, and train our tabular PRM via supervised fine-tuning followed by reinforcement learning with tool-grounded reward shaping. We provide both theoretical analyses and empirical evaluations on the efficacy of our method. Across five challenging tabular reasoning benchmarks, our TATTO-8B PRM achieves an average 30.9% relative gain over the base LRM, consistently surpasses strong baselines such as Qwen-2.5-Math-PRM-72B with up to 9× parameter efficiency, and generalizes robustly across multiple TTS strategies.

1 Introduction

Tabular reasoning has become a fundamental capability for emerging large reasoning models (LRMs), supporting real-world applications such as numerical analysis [1, 48], fact-checking [4, 35, 70], and question answering [52, 36, 25]. Unlike free-form text, tables present information in rows and columns with an implicit relational semi-structure. Reasoning over tables requires both accurate interpretation of tabular content and step-by-step logical inference to generate precise answers [56, 71]. To support such multi-step reasoning, recent advances in test-time scaling (TTS) [34, 71] have substantially enhanced the chain-of-thought (CoT) capabilities of LRMs by leveraging post-training reinforcement learning techniques such as PPO [41] and GRPO [44], aligning model behavior with the demands of complex reasoning tasks. Moreover, works such as the Table-R1 series [60, 65, 21] have extended these advances by transferring reasoning capabilities from general text to the tabular domain.

On the other hand, process reward models (PRMs) [43, 80, 62] have been developed to provide step-level supervision over LRMs' reasoning trajectories, enabling fine-grained guidance to further scale performance at inference time. With increasing compute and emphasis on enhancing LRMs' tabular reasoning abilities [31, 64], a corresponding step-level verifier to evaluate and supervise the reasoning quality of these models is equally important but remains notably absent. This gap motivates our study of a fundamental question: *How can we provide robust step-level supervision to R1-style LRMs in tabular reasoning?*

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

To address this question, we begin by revisiting several advanced PRMs from the general domain and evaluating their ability to provide effective supervision on tabular reasoning tasks. Our analysis reveals that existing PRMs struggle to reliably verify two critical types of table-involved CoT steps: ① *Table Retrieval*, where PRMs fail to supervise whether LRMs extract the correct sub-region of the input table relevant to the query; and ② *Schema Interaction*, where PRMs can't detect attention collapse [10], as LRMs often overlook long-range table dependencies due to inherent locality bias. We also observe that PRMs frequently introduce evaluation errors stemming from table lookup biases or execution mistakes, leading to performance bottlenecks under standard TTS strategies.

Motivated by our preliminary analyses, we introduce **TATTO**, the first table-grounded thinking PRM that leverages tools to provide reward supervision on structured tabular reasoning tasks. In contrast to prior PRMs that overlook table-specific operations, TATTO provides step-level supervision with both table-grounded and model inner-reasoning rewards. In addition, during the verification process, TATTO is able to integrate various types of external tools to engage with table evidence throughout, yielding more precise and reliable supervision. To train our PRM, we first design a scalable data curation pipeline that constructs over 60k high-quality supervision instances by integrating expert verification rationales with programmatic tool executions. We then train our PRM in two stages: (i) supervised fine-tuning to acquire table-aware verification capabilities, and (ii) reinforcement learning with tool-grounded reward shaping to encourage effective tool use and accurate supervision.

To demonstrate the efficacy of TATTO, we provide both theoretical analyses and empirical evaluation. Theoretically, we show that incorporating table-grounded rewards from TATTO yields a strengthened lower-bound performance guarantee on policy improvement. Empirically, across five challenging tabular reasoning benchmarks, incorporating our 8B size TATTO with Best-of-N yields a 30.9% relative gain over the underlying LRMs. In addition, TATTO consistently outperforms other strong PRM baselines such as Qwen-2.5-Math-PRM-72B [72] and GenPRM-32B [73], while using substantially fewer parameters. Our in-depth analyses further show that incorporating RL yields an improvement of 10.2% over supervised fine-tuning alone, and our TATTO generalizes robustly across different TTS strategies such as Beam Search and DVTS.

2 Preliminary

Table Understanding with Reasoning Models. We define a semi-structured table as $T = (H, R)$, where H denotes the set of column headers capturing schema-level semantics, and R denotes the collection of rows, each consisting of cell entries consistently aligned with H. Given a table T and an associated natural language query q, we define a reasoning model as a conditional generation policy $\pi(\tau \mid T, q)$, where $\tau = \{a_1, \ldots, a_i\}$. Here, τ denotes the trajectory sequence of the model response, including both intermediate reasoning steps $a_{\leq L-1}$ and the final answer a_i. In practice, the intermediate reasoning steps can comprise both model-generated reasoning processes and tool-integrated programs that directly operate over the table to retrieve or compute intermediate results. The final answer can take different formats depending on the query type, such as textual/numerical values, boolean outputs (e.g., True/False), or executable programs (e.g., Python, SQL).

Process Reward Model. We denote a PRM as $\mathcal{R}_\theta$, parameterized by θ. At test-time, a PRM is input with a table T, a query q, and a candidate response τ generated by the policy model. The PRM assigns step-level rewards r_i to reflect the correctness of each intermediate CoT step. In table-based tasks, each CoT step typically involves not only the model's internal reasoning but also table-grounded operations (e.g., retrieving a cell value from the input table). Accordingly, we decompose the PRM's step-level reward into two components:

$$r_i = r_{i,\text{rea}} + r_{i,\text{tab}} \quad \text{and} \quad r_\tau = \frac{1}{L}\sum_{i=1}^{L} r_i, \tag{1}$$

where $r_{i,\text{rea}}$ captures the correctness of the reasoning process, $r_{i,\text{tab}}$ reflects the accuracy of table-grounded operations, and r_τ denotes the trajectory-level reward. These process rewards can be further leveraged by an inference-time strategy ϕ to guide resampling, refinement, or candidate selection among the responses generated by the policy model [46].

3 Why Table Reasoning Requires Verifiers Beyond Current PRMs?

To examine the bottlenecks of applying LRMs in tabular reasoning and the effectiveness of existing PRMs on table-involved responses, we first conduct a pilot study on two key questions:

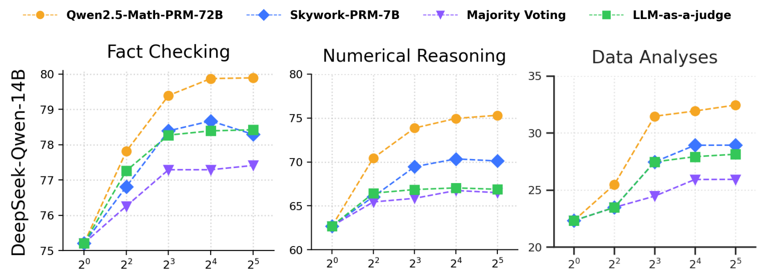

Figure 1: Best-of-N performance of DeepSeek-R1-Distill-Qwen-14B across 3 table tasks with different types of verifiers.

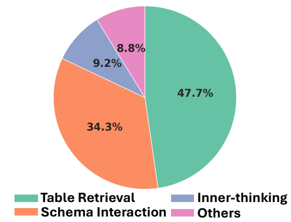

Figure 2: Error Distribution over 4 step categories across 500 incorrect cases after Best-of-N selection.

Table 1: Representative error cases in 4 different reasoning step categories. Each example highlights the erroneous model step in red, the corresponding error description, and the PRM's (mis)judgment reward, illustrating where existing PRMs fail to detect mistakes.

Category	**Wrong Model Answer**	**Error Description & PRM output**
Table Retrieval	<think> Step 1: I need to figure out the average prominence of mountain peaks in the Democratic Republic of the Congo that have an elevation of at least 3000 meters. Step 2: First, I'll review the provided table. My main focus is on the 'country' and 'elevation (m)' columns. *(Wrong Step)* . . . </think> Final Answer is `\boxed{2306.65}`	The model initially retrieves the wrong part of the table (columns *'Country'* and *'Elevation (m)'*), which leads to errors in all subsequent reasoning steps. **PRM** → **{correct}** ✗ Fail to detect the error
Schema Interaction	<think> Step 1: Let me start by looking at the table provided . . . The retrieved columns are . . . Step 4: I'll go through each row and note down the episodes: Episode 1: Rating - 5.1, Episode 2: Rating - 5.2 . . . *(Wrong Step)* Step 5: Adding the rating together is 5.1 + 5.2 + . . . </think> Final Answer is: `\boxed{11.52}`	The model identifies the correct column for the answer, but fails to extract all elements, omitting the last cell. This omission leads to incorrect reasoning in all subsequent steps. **PRM** → **{correct}** ✗ Fail to detect the error
Inner-thinking	<think> Step 1: Based on the table contents, I need to count podiums per season. 2008: $(1+1+1)/3 = 1$, 2009: 6, 2010: 0, 2011: $(2+0)/2 = 1$ . . . Step 4: Then I will average across seasons. Final average is $(1+6+0+1)/4 = 2$ *(Wrong Step)* </think> Final Answer is: `\boxed{2}`	The model incorrectly does the calculation by averaging the season-level means, giving each season equal weight, instead of averaging across all team-seasons. **PRM** → **{incorrect}** ✓ Detect the error
Others	<think> . . . The nation with a total of 13 medals is South Korea.</think> Final Answer is `southkorea` *(Wrong Step)*	The output format is incorrect by not inserting the final answer into `\boxed{}` and cannot be extracted later. **PRM** → **{incorrect}** ✓ Detect the error

***RQ1* -** Beyond free-form text inputs, can common step-level verifiers combined with TTS strategies also enhance the performance of LRMs on tabular reasoning tasks?
***RQ2* -** When reward supervision is required for tabular reasoning tasks, how should PRMs effectively evaluate and guide the quality of each generated reasoning step by LRMs?

For RQ1, we evaluate a widely utilized reasoning model, DeepSeek-R1-Distill-Qwen-14B [13] on TableBench [59], which includes three fundamental table tasks: Fact Checking, Numerical Reasoning, and Data Analyses. As the TTS strategy, we adopt Best-of-N with different feedback verifiers, including two advanced PRMs [72, 14], majority voting [31], and LLM-as-a-judge [75]. Figure 1 reports Best-of-N results across three table reasoning tasks. Incorporating feedback verifiers into Best-of-N indeed improves performance over single-shot generation, with PRMs generally yielding the largest gains. These findings align with prior studies applying PRMs with TTS to other domains such as mathematical reasoning [43, 31, 23]. However, we find that once the number of generated responses surpasses a threshold ($N \geq 8$), the performance all converges to a bottleneck. For instance, the performance of Qwen2.5-Math-PRM-72B on fact checking is 79.19%, 79.82%, and 79.84% for N = $\{8, 16, 32\}$, indicating that further increases in N yield negligible improvements.

> **Observation 1 (Limitation on TTS):** Existing verifiers improve LRMs on tabular reasoning, but their effectiveness quickly saturates, failing to fully exploit additional test-time compute.

Error Analysis. Building on this observation, we further investigate the underlying causes of the performance bottleneck. We conduct an error analysis on both LRM's generation and PRM's supervision processes. Specifically, we sample 500 erroneous responses from the LRM after Best-of-N selection with Qwen2.5-Math-PRM-72B, and ask human experts to carefully classify them into 13 predefined error types (See Appendix C). We then align these error types with 4 broader

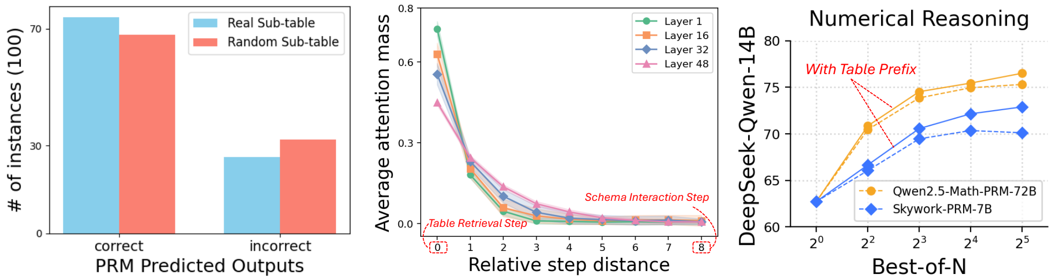

Figure 3: **Left:** PRM judgments on 100 reasoning steps with the real-retrieved/randomly-replaced sub-table. **Middle:** Layer-wise average attention mass vs. relative step distance in tabular reasoning. Attention concentrates on nearby steps, with sharp decay as distance increases. **Right:** Comparison of Best-of-N results on numerical reasoning with/without the table prefix.

reasoning-step categories that reflect the typical flow of an LRM's reasoning process: *Table Retrieval Step* (locating relevant rows/columns regarding the input query), *Schema Interaction Step* (reasoning over the retrieved table contents), *inner-thinking Step* (models' inner reasoning independent of table contents), and *Others* (initial setup or the final output steps). Figure 2 presents the error distribution across 4 reasoning step categories. We find that most errors arise in *Table Retrieval* (47.7%) and *Schema Interaction* (34.3%), implying that PRMs perform reasonably well on independent reasoning but fall short when reasoning steps involve table-specific operations. For better demonstration, we provide representative examples for each category in Table 1.

Why do PRMs fail on table-involved reasoning steps? We next analyze why PRMs lose their supervisory effectiveness when reasoning steps involve table operations. Regarding *table retrieval*, we conduct a contrastive experiment on 500 randomly sampled LRM's output responses by (i) keeping the original retrieved sub-table and (ii) replacing it with a random sub-table region. We compare the PRM's output rewards on these two variants. Figure 3 (left) shows PRM judgments on the table retrieval steps. The identical distributions between real and random sub-tables indicate that PRMs fail to distinguish table retrieval correctness. This suggests that existing PRMs are unable to evaluate if the retrieved portion of the table correctly corresponds to the query.

> **Takeaway 1 (Table Retrieval):** Existing PRMs are insensitive to table retrieval correctness in the reasoning steps and fail to recognize whether the retrieved content corresponds to the query.

Regarding *Schema Interaction*, we observe that in the output trajectories of LRMs, the table retrieval step typically occurs at the beginning, as the model must first extract relevant information from the table to answer the query. In contrast, schema interaction steps are not always adjacent to retrieval; LRMs often perform inner reasoning between retrieval and interaction. As a result, schema interaction steps may occur far from the original table retrieval step. Figure 3 (middle) illustrates the attention distribution for a schema interaction step (step 8) relative to the retrieval step (step 0). Since LRM is auto-regressive, the schema interaction step allocates most of its attention to nearby neighbor steps and little to the earlier retrieval step. Such inherent locality bias leads the model to frequently misinterpret or lose previously retrieved information, though the table retrieval step has already extracted relevant information. Additionally, PRMs fail to detect such misinterpretation, as their judgments are localized to the current step rather than capturing dependency on distant prior steps.

> **Takeaway 2 (Schema Interaction):** Schema interaction steps under-attend to distant table retrieval contents due to locality bias. PRMs miss these failures because they can't look ahead and capture long-range dependencies among distant steps.

Table Prefix is the Key. To address the limitation above, we start by trying a simple input modification of PRMs: prepend the retrieved table contents as a prefix to each schema interaction step before feeding it into PRMs, which provides direct access to the retrieval context instead of long-range dependencies. Figure 3 (right) reports the results with and without the table prefix. Surprisingly, adding the prefix improves the performance of PRMs, suggesting that incorporating retrieval table information as a prefix provides stronger supervision.

Motivation. Our findings above highlight the need for a more robust PRM verifier capable of evaluating both table-involved operations and model inner reasoning. Motivated by this, we design and train a new PRM tailored for tabular reasoning.

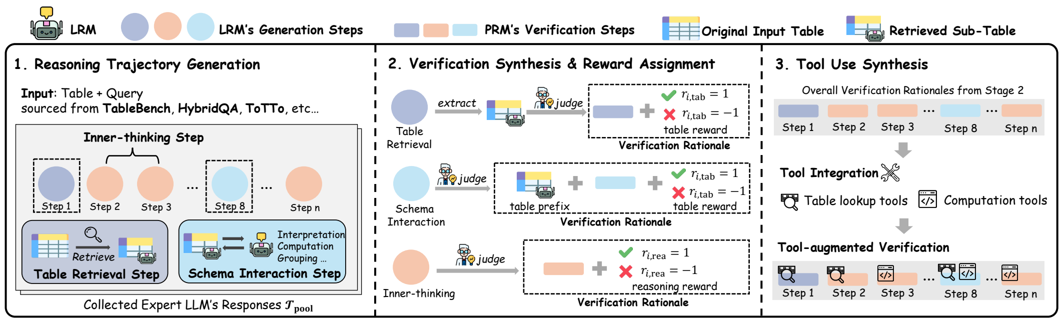

Figure 4: Overview of our data curation pipeline for training TATTO (Detailed in Section 4.1). We treat LRM-generated reasoning steps separately, synthesize verification rationales with assigned step-level rewards, and further augment verification rationales with tool executions for precise supervision.

4 Building a Table-Grounded Step Verifier

We introduce TATTO, a generative PRM that integrates model reasoning with tool-use during its step-by-step verification process to enable effective supervision on both table operations and reasoning steps. In Section 4.1, we first introduce a large-scale agentic data synthesis pipeline designed for PRM training. We then elaborate how TATTO is trained via our two-stage training paradigms in Section 4.2. In Section 4.3, we further provide a theoretical guarantee on TATTO's policy improvement.

4.1 Data Curation Pipline

We find in Section 3 that a major drawback of PRMs is their inability to supervise the table operations, such as retrieval and interaction. Recent studies [40, 11, 38] have advanced LLM agents in autonomously using **tools** to interact with external environments and iteratively refining their actions through reasoning. With several existing table tools available, we incorporate them into the training data so that PRMs can learn to leverage tool execution outputs for more accurate step verification. To this end, we design a comprehensive data curation pipeline that simulates real-world scenarios of PRM tool use and step verification at scale. As illustrated in Figure 4, there are three main stages:

Reasoning Trajectory Generation. We begin by collecting CoT reasoning trajectories generated by expert LRMs (DeepSeek-R1 and Claude Opus 4.1) on various tabular tasks. We sample from a broad range of sources, including TableBench [59], HybridQA [5], ToTTo [35], and WikiTQ [36]. We generate multiple model responses per query, capturing both correct and incorrect reasoning patterns. We then adopt a dual-verification procedure [11], where both human annotators and expert LLMs are employed to examine and filter out low-quality or incomplete CoT data. Through this, we receive a high-quality set of LRMs' output responses $\mathcal{T}_{\text{pool}}$ for subsequent data labeling.

Verification Synthesis & Reward Assignment. Our next step is to provide step-level verification rationales and assign PRM step-reward labels for each candidate response in $\mathcal{T}_{\text{pool}}$. To this end, we first identify the table retrieval and schema interaction steps within each response in $\mathcal{T}_{\text{pool}}$:

Table retrieval steps - We first extract the retrieved sub-table from each step. Then we apply LLM-as-a-judge to evaluate whether retrieved contents are accurate and provide complete rationales for the judgment. We assign step-level table reward $r_{i,\text{tab}} \in \{-1, 1\}$ (in Eq. 1) based on the correctness of the retrieval, while setting $r_{i,\text{rea}}$ to 0. This reward supervision explicitly trains PRMs to recognize if the retrieved sub-table aligns with the input query, addressing the limitation shown in *Takeaway 1*.

Schema interaction steps - We collect the sub-table retrieved from the preceding table retrieval step and use it as a table prefix. If the retrieval is incorrect, we manually replace it with the correct sub-table corresponding to the query. We then prepend this table prefix to the verification rationale generated by LLM-as-a-judge. Finally, we assign the PRM's step-level table reward $r_{i,\text{tab}} \in \{-1, 1\}$ based on the correctness of the schema interaction, and $r_{i,\text{rea}}$ to 0. By explicitly attaching the retrieved sub-table to each schema interaction step, we mitigate the dependencies issue noted in *Takeaway 2*.

Other steps without table operations involved - We directly query an expert LLM (DeepSeek-R1) to generate verification rationales. We assign the PRM's step-level reasoning reward $r_{i,\text{rea}} \in \{-1, 1\}$ based on the correctness of the reasoning, while setting the table reward $r_{i,\text{tab}}$ to 0.

Tool Use Synthesis. To help PRMs learn to leverage tools for more accurate verification, we augment the collected verification rationales by incorporating tool invocation, execution, and feedback into the verification steps. Specifically, whenever the model's inner reasoning involves a calculation or table lookup operation, we replace it with the corresponding tool call and its execution result. We primarily employ two types of tools:

Computation tools - Applying Python or SQL code snippets for arithmetic or aggregation operations. E.g., if a step verifies the sum of a table column, we replace the model's manual calculation with a code snippet that executes the summation and returns the result.

Table lookup tools - Locating and extracting specific rows, columns, or cells from the table. E.g., if a step requires referencing a sub-table cell value during the verification, we replace the model's self-extraction with an explicit lookup tool call that retrieves the corresponding entry.

By integrating verification processes with code snippets and real-time interpreter feedback, we construct roughly 60k data for TATTO's verification reasoning and tool usage.

4.2 Training Details

With the data recipe in place, we then train an off-the-shelf model as our PRM's backbone. Specifically, we initialize TATTO with supervised fine-tuning to acquire table-aware verification capabilities, followed by reinforcement learning to encourage effective tool utilization.

Cold-Start SFT. We first finetune our PRM $\mathcal{R}_\theta$ on the curated dataset described in Section 4.1. During SFT, $\mathcal{R}_\theta$ is trained auto-regressively to learn (i) identifying accurate sub-table regions, (ii) prepending the table prefix to each schema interaction step, and (iii) generating faithful verification rationales that align with the step-level reward labels.

Tool-Grounded Reward Shaping in RL. Previous generative PRMs training [31, 23, 73] typically ended with SFT phase. Recent advances in agentic RL [19, 13] further leverage policy optimization to better align the model's reasoning process with tool invocation and utilization. To enable our PRM to better learn how to dynamically and effectively integrate tools during verification, we optimize $\mathcal{R}_\theta$ via a modified GRPO [44]. Below, we elaborate on how we convert step-level labels y_i into dense, tool-grounded rewards used for policy training.

Specifically, during RL rollouts of each training instance (T, q, τ), we substitute the original rule-based GRPO reward with a newly designed per-step dense reward signal that incorporates label-matching accuracy, confidence calibration, and tool-grounding, i.e.

$$r_i^{\mathrm{GRPO}} = \underbrace{\mathbb{1}\{\hat{y}_i = y_i\}}_{\textbf{label-matching}} - \underbrace{\lambda_{\mathrm{cal}}\Big(-\log \mathcal{R}_\theta(y_i \mid T, q, \tau)\Big)}_{\textbf{confidence calibration}} + \underbrace{\lambda_{\mathrm{tool}} \cdot \mathrm{support}(v_i)}_{\textbf{tool-grounding}}, \tag{2}$$

where v_i denotes the verification rationale associated with step i, corresponding to $a_i \in \tau$, $\mathrm{support}(v_i) \in \{0, 1\}$ measures whether the rationale correctly incorporates tool outputs, and $\lambda_{\mathrm{cal}}, \lambda_{\mathrm{tool}}$ are two tunable coefficients. Beyond training our PRM on step-level correctness via the *label-matching term*, we also incorporate a *confidence calibration term* to encourage the model to place higher probability on correct labels for stable training, and a *tool-grounding term* to promote rationales that correctly leverage tool outputs.

During RL training, we aggregate the per-step rewards r_i^{GRPO} into a trajectory-level reward. These trajectory-level rewards are then normalized within each sampled group to compute group-relative advantages, which are used to update the PRM $\mathcal{R}_\theta$ under the GRPO objective. Note that our newly designed reward can also be directly incorporated into recently advanced GRPO variants, such as GSPO [74] and ARPO [9].

4.3 Inference-time Policy Improvement Guarantee

To elucidate the role of TATTO with its dual rewards on both table operations and reasoning processes (Eq.1) in enhancing the inference-time performance of LRMs, we present a theoretical performance guarantee below on the policy improvement induced by our PRM $\mathcal{R}_\theta$.

Recall that the goal of our PRM is to improve the generated trajectory τ sampled from the policy model π, i.e., $\tau \sim \pi(\cdot \mid T, q)$. By combining the input prefix (T, q), we treat $(T, q, a_1, \ldots, a_{i-1})$ as the current state $\mathbf{s}_i$. At step i, given the current state, the policy model samples an action $a_i \sim \pi(\cdot \mid \mathbf{s}_i)$. The Q-value of policy π for the state–action pair $(\mathbf{s}_i, a_i)$ is defined as the expected future success:

$$Q^\pi(\mathbf{s}_i, a_i) = Q^\pi\big((T, q, a_1, \dots, a_{i-1}), a_i\big) = \mathbb{E}_{a_{i+1},\dots,a_L \sim \pi(\cdot|\mathbf{s}_i)}\left[\mathbb{1}_{a_L \text{ is correct}}\right]. \tag{3}$$

Here, we use the Q-value to measure the success probability of reaching the final correct answer. Similarly, the value at state $\mathbf{s}_i$ can be defined as the expectation Q-values over the next action, i.e., $V^\pi(\mathbf{s}_i) = \mathbb{E}_{a_i \sim \pi(\cdot|\mathbf{s}_i)}[Q^\pi(\mathbf{s}_i, a_i)]$. In Theorem 4.1, we establish a lower-bound guarantee on policy improvement, in terms of $V^\pi(\mathbf{s}_i)$, after a single step of natural policy gradient update when incorporating the process reward supervision r_i from our PRM $\mathcal{R}_\theta$.

Theorem 4.1 (**Policy Improvement Guarantee (Lower Bound)**). *Let π denote the current policy. After one step of natural policy gradient update guided by the PRM $\mathcal{R}_\theta$, the updated policy is $\pi'(a_i \mid \mathbf{s}_i) \propto \exp(Q^\pi(\mathbf{s}_i, a_i) + r_i(\mathbf{s}_i, a_i))$, where $r_i(\mathbf{s}_i, a_i) = r_{i,rea}(\mathbf{s}_i, a_i) + r_{i,tab}(\mathbf{s}_i, a_i)$ (see Eq. 1). Then, the expected improvement over the state distribution ρ satisfies:*

$$\mathbb{E}_{\mathbf{s}_i \sim \rho}\left[V^{\pi'}(\mathbf{s}_i) - V^\pi(\mathbf{s}_i)\right] \gtrsim \underbrace{\mathbb{E}_{\mathbf{s}_i \sim \rho}\mathrm{Var}_{a_i \sim \pi(\cdot|\mathbf{s}_i)}\left[r_{i,rea}(\mathbf{s}_i, a_i)\right]}_{\textit{distinguishability from reasoning reward}} + \underbrace{\mathbb{E}_{\mathbf{s}_i \sim \rho}\mathrm{Var}_{a_i \sim \pi(\cdot|\mathbf{s}_i)}\left[r_{i,tab}(\mathbf{s}_i, a_i)\right]}_{\textit{distinguishability from table reward}}$$
$$+ \underbrace{\mathbb{E}_{\mathbf{s}_i \sim \rho}\mathbb{E}_{a_i \sim \pi(\cdot|\mathbf{s}_i)}\left[r_i(\mathbf{s}_i, a_i) A^\pi(\mathbf{s}_i, a_i)\right]}_{\textit{alignment between overall process reward } r_i \textit{ and } A^\pi}, \tag{4}$$

where $A^\pi(\mathbf{s}_i, a_i) = Q^\pi(\mathbf{s}_i, a_i) - V^\pi(\mathbf{s}_i)$ denotes the advantage of π for the state–action pair $(\mathbf{s}_i, a_i)$.

Remark 4.2. The variance term $\mathbb{E}_{\mathbf{s}_i \sim \rho}\mathrm{Var}_{a_i \sim \pi(\cdot|\mathbf{s}_i)}[r_{i,\text{tab}}(\mathbf{s}_i, a_i)] \geq 0$, introduced through our PRM's table reward $r_{i,\text{tab}}$, yields an improved lower-bound guarantee compared to common PRMs that rely solely on the reasoning reward $r_{i,\text{rea}}$.

Theorem 4.1 verifies our observation in Section 3 that incorporating additional reward supervision on table operation steps during policy models (LRMs) generation can further enhance their inference-time performance. The complete proof is presented in Appendix B. In the following section, we further empirically evaluate the effectiveness of our TATTO across various tabular reasoning tasks.

5 Empirical Evaluations

Baselines and Models. We compare TATTO against various types of verifiers, including advanced PRMs, majority voting [31], and LLM-as-a-judge [75]. The setups for these baselines are aligned with Section 3. For PRMs, we include both discriminative (Qwen-PRM series [72], Math-Shepherd-PRM [54], and Skywork-PRM [14]) and generative (ThinkPRM [23] and GenPRM [73]). Regarding the policy reasoning models, we evaluate our proposed method on DeepSeek-R1-Distill-Qwen-14B [13]. Further details on the baselines and policy models setups are provided in Appendix D.1.

Datasets. We evaluate on four representative and challenging benchmarks spanning diverse tabular reasoning tasks: (i) TableBench (TB) [59], a recently released benchmark for complex table reasoning with 886 test cases across 18 categories; we focus on three core sub-tasks: Numerical Reasoning (NR), Fact Checking (FC), and Data Analysis (DA). (ii) WTQ [36], a benchmark for complex question answering over Wikipedia tables. (iii) MMQA [57], a multi-table understanding benchmark covering retrieval, text-to-SQL generation, multi-table QA, and key selection.

Implementation Details. We train TATTO on the off-the-shelf Qwen-3-8B model [61] using our curated 60k training dataset described in Section 4.1. All training and inference experiments are conducted on 8×A100-80G GPUs. For the TTS strategy, we adopt three representative methods to evaluate the effectiveness of TATTO under inference scaling: Best-of-N [3], Beam Search [46], and Diverse Verifier Tree Search (DVTS) [2]. Additional details, including TATTO training setup and the configurations of the three TTS strategies, are provided in Appendix D.2.

5.1 Main Results

Table 2 reports the best-of-N performance on the DeepSeek-R1-Distill-Qwen-14B policy model across 5 table reasoning tasks. Notably, TATTO consistently outperforms strong baselines such as GenPRM (32B) and Qwen-Math-PRM (72B) with only 8B parameter size. For example, on TB-DA, a particularly challenging task for math-oriented PRMs, TATTO achieves the largest relative gain,

Table 2: Main results of TATTO on 5 different tabular reasoning tasks. We report the best-of-N performance using DeepSeek-R1-Distill-Qwen-14B as the policy model and compare against various feedback verifiers. The best and second-best results are highlighted. TATTO consistently achieves state-of-the-art TTS performance with significantly fewer parameters.

Verifer (Best-of-N)	Params	TB-NR				TB-FC				TB-DA				WTQ				MMQA			
		4	8	16	32	4	8	16	32	4	8	16	32	4	8	16	32	4	8	16	32
Majority Vote	-	65.5	65.9	66.8	66.5	76.2	77.3	77.3	77.4	23.5	24.5	26.0	26.1	64.7	65.3	67.3	67.0	18.4	19.4	20.4	20.1
LLM-as-a-judge	-	66.7	66.9	67.1	66.9	77.2	78.3	78.4	78.6	23.5	27.4	28.0	28.4	65.2	66.4	68.1	68.1	19.6	21.3	22.5	22.7
Skywork-PRM-7B	7B	66.1	69.5	70.3	70.1	76.8	78.4	78.6	78.3	24.1	27.5	28.9	29.1	65.9	67.5	68.4	68.6	21.4	24.6	25.1	25.3
Math-Shepherd-PRM-7B	7B	67.2	70.6	71.5	71.8	76.2	76.9	76.8	77.1	22.7	24.8	26.4	25.9	66.8	68.7	69.6	69.3	22.0	25.2	25.9	26.1
Qwen-2.5-Math-PRM-7B	7B	66.9	70.1	71.7	72.5	75.4	77.2	77.9	77.4	23.2	25.4	26.3	26.6	65.2	68.5	69.6	69.7	23.5	25.2	27.1	27.3
ThinkPRM	14B	69.2	70.7	73.5	73.8	75.8	75.4	76.3	76.9	21.6	22.7	23.1	22.8	64.3	66.1	65.7	65.9	22.4	22.7	23.6	23.0
GenPRM	32B	**71.5**	73.5	73.7	74.2	76.3	78.5	79.2	79.4	25.3	27.9	30.2	30.7	**69.8**	**72.5**	73.3	73.1	23.8	25.4	26.2	26.4
Qwen-2.5-Math-PRM-72B	72B	70.4	73.8	74.9	75.3	**77.8**	79.2	79.8	79.8	25.5	31.5	32.0	32.4	69.2	71.8	73.0	72.6	24.4	26.8	28.7	28.6
TATTO	8B	71.2	**74.2**	**76.4**	**78.1**	77.4	**79.6**	**81.2**	**82.0**	**27.7**	**31.9**	**33.6**	**34.3**	**69.8**	72.3	**73.5**	**74.9**	**25.1**	**27.2**	**29.1**	**30.5**

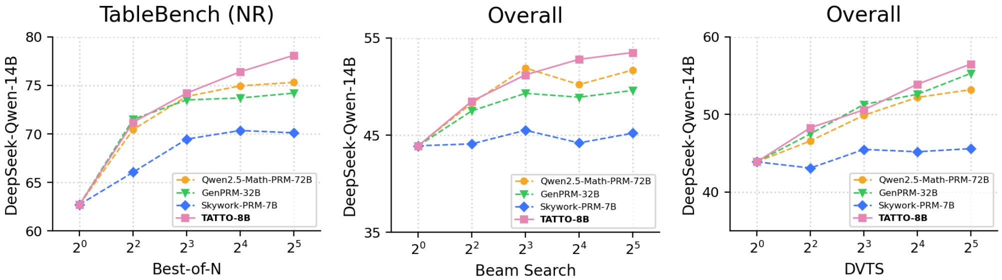

Figure 5: Performance of TATTO under three TTS strategies. Left: Best-of-N on TableBench-NR, where TATTO achieves consistent improvements as N increases. Middle: Average results on Beam Search across five table reasoning tasks. Right: Average results on Diverse Verifier Tree Search (DVTS) across the same tasks.

improving from 27.7% at N= 4 to 34.3% at N= 32. In addition, as observed earlier in Section 3, existing PRMs often saturate or yield suboptimal performance beyond a threshold of N. In contrast, TATTO continues to deliver consistent improvements as the response group size increases.

Figure 5 (left) visualizes the performance gains with increasing N on TB-NR. These results highlight (i) the effectiveness and relevance of our curated training dataset, and (ii) the ability of our method to deliver stronger table-grounded reward supervision, achieving substantial accuracy gains while being up to 9× more parameter-efficient than the strongest baselines.

5.2 In-depth Analyses on TATTO

Generalizability on Other TTS Strategies. Beyond Best-of-N, we also evaluate TATTO on two additional TTS strategies: beam search and DVTS. Figure 5 (middle and right) shows the average performance across the 5 tabular reasoning tasks. Across both strategies compared to other PRM baselines, TATTO consistently delivers steady improvements with the increasing budgets on N. For example, in beam search, TATTO improves from 45.0% to 54.8%, whereas GenPRM plateaus around 51% and Skywork-PRM remains below 46%. These results demonstrate the robust generalizability of our method across various TTS strategies.

Mastery of RL with Bootstrapping from SFT. To investigate tool integration and reasoning capabilities of TATTO through RL training, we conduct additional experiments to compare the performance before (cold-start SFT only) and after RL training on TableBench. Table 3 (first two rows) reports the Best-of-N results for the two training stages of TATTO. After RL training, the overall performance improves by 10.2% relative to the SFT stage.

We further present a case study illustrating the difference between the verification processes at the two training stages on a specific instance in Figure 7 (Appendix F). When facing the same step (Step 3), the SFT-stage relies on inner text reasoning to verify the calculation, but introduces numerical errors that lead to incorrect justification of the step's correctness. In contrast, the RL-stage learns to leverage the computation tool with concise Python code, ensuring accurate calculations and thereby providing more reliable reward supervision on the policy model's responses. In addition, we randomly sample 500 reasoning trajectories from both stages of TATTO on the same set of inputs and observe a 26.3%

Table 3: In-depth analysis of TATTO on TableBench. We compare performance across SFT and RL training stages, along with ablations on reward design.

TATTO	TB-NR				TB-FC				TB-DA			
	4	8	16	32	4	8	16	32	4	8	16	32
SFT only	67.9	69.1	72.0	73.7	71.5	73.0	74.6	75.2	23.3	25.6	26.2	26.4
ours (SFT+RL)	**71.2**	**74.2**	**76.4**	**78.1**	**77.4**	**79.6**	**81.2**	**82.0**	**27.7**	**31.9**	**33.6**	**34.3**
rule-based	67.0	68.4	70.4	73.1	71.6	74.0	74.9	75.8	25.5	27.4	28.0	28.6
w/o tool-grounding	68.5	71.1	72.7	74.6	73.2	75.6	75.5	76.3	26.2	28.1	28.7	30.3
w/o confidence calibration	71.1	73.7	74.3	76.2	76.4	76.7	78.4	80.5	27.4	29.5	31.3	33.2

improvement in the tool-integration ratio after RL training, indicating our model learns to utilize tools better for step-level verification during RL rollouts.

Reward Shaping during RL Training. We analyze the contribution of each component in our reward shaping (Eq. 2) during RL training. Table 3 reports the ablation results for each component. Removing the tool-grounding term results in a significant performance drop of 3.9% across all three tasks, e.g., $\downarrow 4\%$ on TB-DA at N = 32, underscoring the necessity of encouraging tool utilization during training. Similarly, excluding the confidence calibration term degrades performance by 1.6% on average, indicating that calibrated probability assignment provides a complementary role in stabilizing reward signals. We also investigate the original rule-based group-relative reward from GRPO, which yields marginal improvement after SFT. This indicates that solely relying on the original reward (designed primarily for policy model training) does not transfer well when applying RL to a reward model. Due to page limits, we leave additional ablation studies in Appendix E.

6 Related Works

Reasoning over semi-structured tables poses a unique challenge for LLMs, requiring them to bridge natural language understanding with structured reasoning over rows, columns, and cell values [20, 71]. Recent works [50, 18, 8, 15] have investigated tabular reasoning on several downstream tasks, including table QA [49, 36, 5, 79], table fact verification [4, 35, 59], text-to-SQL [33], etc.

Early-stage methods, such as TAPAS [17] and TaBERT [66], encode table data into transformer-based encoder representations to support end-to-end table understanding. Later studies leverage the capabilities of LLMs to apply either prompt engineering [48, 49, 56] or supervised fine-tuning techniques [26, 47, 69] for enhanced reasoning on tables, achieving stronger generalization and adaptability across diverse tasks. More recent works, including the Table-R1 series [60, 65, 21] and Reasoning-Table [24], leverages post-training RL methods such as GRPO [44] to acquire higher-quality reasoning paths during reasoning over table information.

While these recent advances have focused on improving the generation ability of models on tables, how to provide robust and verifiable reward supervision for the lengthy and complex output trajectories generated by the table-specific reasoning models remains largely unexplored. This essential yet overlooked gap motivates us to develop the first tool-use and thinking PRM, which is specifically designed and utilized for enhancing test time scaling on tabular reasoning tasks. We leave more detailed discussions on Process Reward Models and Tool Integration with RL in Appendix A.

7 Conclusion

We introduced TATTO, the first tool-augmented thinking PRM tailored for tabular reasoning. By diagnosing why existing verifiers fail on table retrieval and schema interaction, we built a scalable pipeline with expert rationales, table prefixes, and tool-augmented verification, and trained our model via SFT followed by RL with reward shaping. Theoretically, TATTO strengthens policy improvement guarantees by supervising both reasoning and table-grounded operations in each reasoning step. Empirically, TATTO achieves state-of-the-art performance across five table benchmarks, surpassing strong PRMs with up to 9× parameter efficiency and generalizing across multiple TTS strategies. Our results underscore the importance of table-grounded reward supervision and point toward future directions in reward modeling for structured reasoning tasks. We leave additional discussion of limitations and broader impacts to Appendix G.

References

[1] Mubashara Akhtar, Abhilash Shankarampeta, Vivek Gupta, Arpit Patil, Oana Cocarascu, and Elena Simperl. Exploring the numerical reasoning capabilities of language models: A comprehensive analysis on tabular data. *arXiv preprint arXiv:2311.02216*, 2023.

[2] Edward Beeching, Lewis Tunstall, and Sasha Rush. Scaling test-time compute with open models.

[3] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling, 2024.

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

[5] Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Wang. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. *arXiv preprint arXiv:2004.07347*, 2020.

[6] Xiusi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang, Denghui Zhang, Tong Zhang, Hanghang Tong, and Heng Ji. Rm-r1: Reward modeling as reasoning, 2025.

[7] Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

[8] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. Turl: Table understanding through representation learning. *ACM SIGMOD Record*, 51(1):33–40, 2022.

[9] Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, et al. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*, 2025.

[10] Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International conference on machine learning*, pages 2793–2803. PMLR, 2021.

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

[12] Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*, 2025.

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

[14] Jujie He, Tianwen Wei, Rui Yan, Jiacai Liu, Chaojie Wang, Yimeng Gan, Shiwen Tu, Chris Yuhao Liu, Liang Zeng, Xiaokun Wang, Boyang Wang, Yongcong Li, Fuxiang Zhang, Jiacheng Xu, Bo An, Yang Liu, and Yahui Zhou. Skywork-o1 open series. `https://huggingface.co/Skywork`, November 2024.

[15] Xinrui He, Yikun Ban, Jiaru Zou, Tianxin Wei, Curtiss B Cook, and Jingrui He. Llm-forest: Ensemble learning of llms with graph-augmented prompts for data imputation. *arXiv preprint arXiv:2410.21520*, 2024.

[16] Xinyi He, Jiaru Zou, Yun Lin, Mengyu Zhou, Shi Han, Zejian Yuan, and Dongmei Zhang. CoCoST: Automatic complex code generation with online searching and correctness testing. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19433–19451, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[17] Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. Tapas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349*, 2020.

[18] Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. Tabbie: Pretrained representations of tabular data. *arXiv preprint arXiv:2105.02584*, 2021.

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

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

[21] Rihui Jin, Zheyu Xin, Xing Xie, Zuoyi Li, Guilin Qi, Yongrui Chen, Xinbang Dai, Tongtong Wu, and Gholamreza Haffari. Table-r1: Self-supervised and reinforcement learning for program-based table reasoning in small language models. *arXiv preprint arXiv:2506.06137*, 2025.

[22] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the nineteenth international conference on machine learning*, pages 267–274, 2002.

[23] Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. Process reward models that think, 2025.

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

[25] Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. Table-gpt: Table-tuned gpt for diverse table tasks. *arXiv preprint arXiv:2310.09263*, 2023.

[26] Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. Table-gpt: Table fine-tuned gpt for diverse table tasks. *Proceedings of the ACM on Management of Data*, 2(3):1–28, 2024.

[27] Wendi Li and Yixuan Li. Process reward model with q-value rankings. *arXiv preprint arXiv:2410.11287*, 2024.

[28] Yinghui Li, Zishan Xu, Shaoshen Chen, Haojing Huang, Yangning Li, Yong Jiang, Zhongli Li, Qingyu Zhou, Hai-Tao Zheng, and Ying Shen. Towards real-world writing assistance: A chinese character checking benchmark with faked and misspelled characters. *CoRR*, abs/2311.11268, 2023.

[29] Yinghui Li, Qingyu Zhou, Yuanzhen Luo, Shirong Ma, Yangning Li, Hai-Tao Zheng, Xuming Hu, and Philip S. Yu. When llms meet cunning questions: A fallacy understanding benchmark for large language models. *CoRR*, abs/2402.11100, 2024.

[30] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024.

[31] Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. Can 1b llm surpass 405b llm? rethinking compute-optimal test-time scaling. *arXiv preprint arXiv:2502.06703*, 2025.

[32] Maxwell-Jia. AIME 2024 dataset. `https://huggingface.co/datasets/Maxwell-Jia/AIME_2024`, 2024. Accessed: 2025-05-15.

[33] Ali Mohammadjafari, Anthony S Maida, and Raju Gottumukkala. From natural language to sql: Review of llm-based text-to-sql systems. *arXiv preprint arXiv:2410.01066*, 2024.

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

[35] Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. Totto: A controlled table-to-text generation dataset. In *EMNLP 2020*, pages 1173–1186, 2020.

[36] Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, 2015.

[37] Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. In Chengqing Zong and Michael Strube, editors, *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China, July 2015. Association for Computational Linguistics.

[38] Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs, 2025.

[39] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023.

[40] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

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

[42] Kwangwook Seo, Donguk Kwon, and Dongha Lee. Mt-raig: Novel benchmark and evaluation framework for retrieval-augmented insight generation over multiple tables. *arXiv preprint arXiv:2502.11735*, 2025.

[43] Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

[44] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. *URL https://arxiv. org/abs/2402.03300*, 2024.

[45] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

[46] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[47] Aofeng Su, Aowen Wang, Chao Ye, Chen Zhou, Ga Zhang, Guangcheng Zhu, Haobo Wang, Haokai Xu, Hao Chen, Haoze Li, et al. Tablegpt2: A large multimodal model with tabular data integration. *arXiv preprint arXiv:2411.02059*, 2024.

[48] Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 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, 2024.

[49] Yuan Sui, Jiaru Zou, Mengyu Zhou, Xinyi He, Lun Du, Shi Han, and Dongmei Zhang. Tap4llm: Table provider on sampling, augmenting, and packing semi-structured data for large language model reasoning. *arXiv preprint arXiv:2312.09039*, 2023.

[50] Nan Tang, Ju Fan, Fangyi Li, Jianhong Tu, Xiaoyong Du, Guoliang Li, Sam Madden, and Mourad Ouzzani. Rpt: relational pre-trained transformer is almost all you need towards democratizing data preparation. *arXiv preprint arXiv:2012.02469*, 2020.

[51] Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback, 2022.

[52] Svitlana Vakulenko and Vadim Savenkov. Tableqa: Question answering on tabular data. *arXiv preprint arXiv:1705.06504*, 2017.

[53] Jun Wang, Meng Fang, Ziyu Wan, Muning Wen, Jiachen Zhu, Anjie Liu, Ziqin Gong, Yan Song, Lei Chen, Lionel M Ni, et al. Openr: An open source framework for advanced reasoning with large language models. *arXiv preprint arXiv:2410.09671*, 2024.

[54] Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations, 2024.

[55] Yinjie Wang, Ling Yang, Ye Tian, Ke Shen, and Mengdi Wang. Co-evolving llm coder and unit tester via reinforcement learning. *arXiv preprint arXiv:2506.03136*, 2025.

[56] 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. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *arXiv preprint arXiv:2401.04398*, 2024.

[57] Jian Wu, Linyi Yang, Dongyuan Li, Yuliang Ji, Manabu Okumura, and Yue Zhang. Mmqa: Evaluating llms with multi-table multi-hop complex questions. In *The Thirteenth International Conference on Learning Representations*, 2025.

[58] Jian Wu, Linyi Yang, Dongyuan Li, Yuliang Ji, Manabu Okumura, and Yue Zhang. Mmqa: Evaluating llms with multi-table multi-hop complex questions. In *The Thirteenth International Conference on Learning Representations*, 2025.

[59] Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, Xinrun Du, Di Liang, Daixin Shu, Xianfu Cheng, Tianzhen Sun, et al. Tablebench: A comprehensive and complex benchmark for table question answering. *arXiv preprint arXiv:2408.09174*, 2024.

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

[61] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report, 2025.

[62] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

[63] Jian Yang, Shuming Ma, Dongdong Zhang, Zhoujun Li, and Ming Zhou. Improving neural machine translation with soft template prediction. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5979–5989. Association for Computational Linguistics, 2020.

[64] Ling Yang, Zhaochen Yu, Bin Cui, and Mengdi Wang. Reasonflux: Hierarchical llm reasoning via scaling thought templates. *arXiv preprint arXiv:2502.06772*, 2025.

[65] Zheyuan Yang, Lyuhao Chen, Arman Cohan, and Yilun Zhao. Table-r1: Inference-time scaling for table reasoning, 2025.

[66] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. Tabert: Pretraining for joint understanding of textual and tabular data. *arXiv preprint arXiv:2005.08314*, 2020.

[67] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *EMNLP 2018*, pages 3911–3921, 2018.

[68] Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: Llm self-training via process reward guided tree search. *arXiv preprint arXiv:2406.03816*, 2024.

[69] Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. Tablellama: Towards open large generalist models for tables. *arXiv preprint arXiv:2311.09206*, 2023.

[70] Xiaokang Zhang, Jing Zhang, Zeyao Ma, Yang Li, Bohan Zhang, Guanlin Li, Zijun Yao, Kangli Xu, Jinchang Zhou, Daniel Zhang-Li, Jifan Yu, Shu Zhao, Juanzi Li, and Jie Tang. Tablellm: Enabling tabular data manipulation by llms in real office usage scenarios. *CoRR*, abs/2403.19318, 2024.

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

[72] Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*, 2025.

[73] Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, and Bowen Zhou. Genprm: Scaling test-time compute of process reward models via generative reasoning, 2025.

[74] Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

[75] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023.

[76] Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand, 2024. Association for Computational Linguistics.

[77] Jialun Zhong, Wei Shen, Yanzeng Li, Songyang Gao, Hua Lu, Yicheng Chen, Yang Zhang, Wei Zhou, Jinjie Gu, and Lei Zou. A comprehensive survey of reward models: Taxonomy, applications, challenges, and future. *arXiv preprint arXiv:2504.12328*, 2025.

[78] Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, 2017.

[79] Jiaru Zou, Dongqi Fu, Sirui Chen, Xinrui He, Zihao Li, Yada Zhu, Jiawei Han, and Jingrui He. Gtr: Graph-table-rag for cross-table question answering. *arXiv preprint arXiv:2504.01346*, 2025.

[80] Jiaru Zou, Ling Yang, Jingwen Gu, Jiahao Qiu, Ke Shen, Jingrui He, and Mengdi Wang. Reasonflux-prm: Trajectory-aware prms for long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2506.18896*, 2025.

Table of Contents

Appendix

A Additional Related Work

Table Question Answering. The evolution of Table Question Answering (Table QA) research [20] has been propelled by the creation of sophisticated evaluation resources that facilitate semantic parsing capabilities [63, 28, 29]. Foundational works, including WTQ [37] and TabFact [4], established initial evaluation paradigms through Wikipedia-derived HTML table QA pairs. Structured supervision has also been explored in alternative benchmarks such as WikiSQL [78] and Spider [67], where logical expressions serve as explicit annotations to encourage systematic reasoning. More recent studies such as MultiTableQA [79], MT-RAIG [42], and MMQA [58] has shifted towards multi-hop reasoning.

PRMs for Test-time Scaling. Process Reward Models (PRMs) [30, 51, 68] deliver fine-grained, step-level feedback to guide model reasoning, assigning intermediate rewards to individual reasoning steps rather than only judging final answers [12, 55, 6]. Prominent PRMs, including Math-Shepherd [54], Skywork-PRM [14], and the Qwen2.5-Math-PRM family [72], are trained using a mix of human annotations and synthesized supervision to score model-generated solution steps across domains such as math [32], scientific reasoning [39], and programming [16]; more recently, Think-PRM proposes a generative verifier to produce long-chain CoT evaluations [23]. PRMs have been incorporated into training-time optimization as reward signals via step-verified online RL and verifier-guided self-training [27, 12, 7], and into inference-time scaling by coupling step-level scoring with search/decoding strategies [73, 23, 64], including beam search, reward-guided tree search, and Best-of-N sampling.

Discriminative vs. Generative PRM. In general, PRMs can be categorized as discriminative and generative evaluators [77]. A **discriminative PRM** treats verification as classification, directly predicting the correctness of each reasoning step with a scalar score. It is typically trained on step-level labels using cross-entropy loss, making it heavily reliant on step-level reward annotations. A **generative PRM** instead frames verification as conditional generation. It is trained with the standard language modeling objective to first generate rationales and then verify each step's correctness via a judgment token (e.g., [correct, incorrect]).

B Proof of Theorem 4.1

Notational conventions. We use $\mathbf{s}_i$ for a state, a_i for an action, π for the current policy, and π' for the updated policy. The advantage is $A^\pi(\mathbf{s}_i, a_i) = Q^\pi(\mathbf{s}_i, a_i) - V^\pi(\mathbf{s}_i)$. The PRM signal at a step is the overall process reward, defined as

$$r_i(\mathbf{s}_i, a_i) \;\triangleq\; r_{i,\text{rea}}(\mathbf{s}_i, a_i) + r_{i,\text{tab}}(\mathbf{s}_i, a_i).$$

For a fixed $\mathbf{s}_i$, we write $\mathbb{E}_\pi[\cdot] \equiv \mathbb{E}_{a_i\sim\pi(\cdot|\mathbf{s}_i)}[\cdot]$, $\mathrm{Var}_\pi[\cdot] \equiv \mathrm{Var}_{a_i\sim\pi(\cdot|\mathbf{s}_i)}[\cdot]$, and $\mathrm{Cov}_\pi(\cdot,\cdot) \equiv \mathrm{Cov}_{a_i\sim\pi(\cdot|\mathbf{s}_i)}(\cdot,\cdot)$. Expectations over states use the subscript explicitly, e.g., $\mathbb{E}_{\mathbf{s}_i\sim\rho}[\cdot]$. We use $d_\rho^{\pi'}$ for the discounted state distribution under π' starting from ρ.

We start the proof by introducing two standard lemmas that will be used repeatedly; both are well-known results in the RL literature, and we omit their proofs here for brevity.

Lemma B.1 (**Performance Difference Lemma (PDL)**). *For any pair of policies π and π' defined over the same Markov decision process with initial state distribution ρ, the following identity holds:*

$$\mathbb{E}_{\mathbf{s}_i\sim\rho}\Big[V^{\pi'}(\mathbf{s}_i) - V^\pi(\mathbf{s}_i)\Big] \;=\; \mathbb{E}_{\mathbf{s}_i\sim d_\rho^{\pi'}}\mathbb{E}_{a_i\sim\pi'(\cdot|\mathbf{s}_i)}\big[A^\pi(\mathbf{s}_i, a_i)\big].$$

See proof of Lemma 6.1 in [22].

Lemma B.2 (**Natural policy gradient (NPG) update form**). *Fix a step size $\gamma > 0$. If the NPG update is guided by the signal $A^\pi(\mathbf{s}_i, a_i) + r_i(\mathbf{s}_i, a_i)$, then*

$$\begin{aligned}
\pi'(a_i \mid \mathbf{s}_i) &\propto\; \pi(a_i \mid \mathbf{s}_i)\, \exp\Big(\gamma\big(A^\pi(\mathbf{s}_i, a_i) + r_i(\mathbf{s}_i, a_i)\big)\Big),\\
Z^\pi(\mathbf{s}_i) &\triangleq\; \mathbb{E}_{a_i\sim\pi(\cdot|\mathbf{s}_i)}\Big[\exp\Big(\gamma\big(A^\pi(\mathbf{s}_i, a_i) + r_i(\mathbf{s}_i, a_i)\big)\Big)\Big],\\
\textit{so that}\quad \frac{\pi'(a_i \mid \mathbf{s}_i)}{\pi(a_i \mid \mathbf{s}_i)} &= \frac{\exp(\cdot)}{Z^\pi(\mathbf{s}_i)}.
\end{aligned} \tag{5}$$

See proof of Lemma F.2 in [43]. Next, we restate Theorem 4.1 in the following proposition.

Proposition B.3 (**Full-strength policy improvement lower bound**). *Let π' be the NPG update in Lemma B.2. There exists $\eta = \Theta(\gamma)$ such that*

$$\begin{aligned}
\mathbb{E}_{\mathbf{s}_i\sim\rho}\Big[V^{\pi'}(\mathbf{s}_i) - V^\pi(\mathbf{s}_i)\Big] \;\gtrsim\; \mathbb{E}_{\mathbf{s}_i\sim\rho}\Big[&\underbrace{\mathrm{Var}_\pi\big[r_{i,\text{rea}}(\mathbf{s}_i, a_i)\big]}_{\textit{distinguishability (reasoning reward)}} + \underbrace{\mathrm{Var}_\pi\big[r_{i,\text{tab}}(\mathbf{s}_i, a_i)\big]}_{\textit{distinguishability (table reward)}}\\
&+2\underbrace{\mathrm{Cov}_\pi\big(r_{i,\text{rea}}(\mathbf{s}_i, a_i), r_{i,\text{tab}}(\mathbf{s}_i, a_i)\big)}_{\textit{alignment between } r_{i,rea} \textit{ and } r_{i,tab}} + \underbrace{\mathbb{E}_\pi\big[r_i(\mathbf{s}_i, a_i)\, A^\pi(\mathbf{s}_i, a_i)\big]}_{\textit{alignment of } r_i \textit{ with } A^\pi}\Big].
\end{aligned} \tag{6}$$

Proof of Proposition B.3. We now combine the performance difference lemma with the NPG update to derive a variance–alignment lower bound, while first retaining the covariance term between the reward components. By Lemma B.1, we have

$$\mathbb{E}_{\mathbf{s}_i\sim\rho}\big[V^{\pi'}(\mathbf{s}_i) - V^\pi(\mathbf{s}_i)\big] \;=\; \mathbb{E}_{\mathbf{s}_i\sim d_\rho^{\pi'}}\mathbb{E}_{a_i\sim\pi'(\cdot|\mathbf{s}_i)}\big[A^\pi(\mathbf{s}_i, a_i)\big]. \tag{7}$$

Exponential tilting and a log-partition bound. Let us define the log-partition at state $\mathbf{s}_i$ by

$$\log Z^\pi(\mathbf{s}_i) \;=\; \log \mathbb{E}_{a_i\sim\pi(\cdot|\mathbf{s}_i)} \exp\Big(\gamma\big(A^\pi(\mathbf{s}_i, a_i) + r_i(\mathbf{s}_i, a_i)\big)\Big).$$

From Lemma B.2, we have

$$A^\pi(\mathbf{s}_i, a_i) = \frac{1}{\gamma}\log\frac{\pi'(a_i \mid \mathbf{s}_i)}{\pi(a_i \mid \mathbf{s}_i)} - r_i(\mathbf{s}_i, a_i) + \frac{1}{\gamma}\log Z^\pi(\mathbf{s}_i).$$

Averaging over $a_i \sim \pi'(\cdot \mid \mathbf{s}_i)$, using $\mathbb{E}_{\pi'}[\log\frac{\pi'}{\pi}] \geq 0$ and Jensen plus $\mathbb{E}_\pi[A^\pi(\mathbf{s}_i, a_i)] = 0$ gives

$$\mathbb{E}_{a_i\sim\pi'(\cdot|\mathbf{s}_i)}[A^\pi(\mathbf{s}_i, a_i)] \;\geq\; -\mathbb{E}_{a_i\sim\pi'(\cdot|\mathbf{s}_i)}[r_i(\mathbf{s}_i, a_i)] + \mathbb{E}_{a_i\sim\pi(\cdot|\mathbf{s}_i)}[r_i(\mathbf{s}_i, a_i)]. \tag{8}$$

Plugging this into Eq. 7 yields the basic inner-product lower bound

$$\mathbb{E}_{\mathbf{s}_i\sim\rho}\big[V^{\pi'}(\mathbf{s}_i)-V^{\pi}(\mathbf{s}_i)\big] \;\geq\; \mathbb{E}_{\mathbf{s}_i\sim d_\rho^{\pi'}}\langle \pi'(\cdot\mid\mathbf{s}_i)-\pi(\cdot\mid\mathbf{s}_i),\, r_i(\mathbf{s}_i,\cdot)\rangle\,. \tag{9}$$

Small-γ expansion of the policy move. For sufficiently small γ, a first-order expansion of the exponential tilt implies

$$\langle \pi'(\cdot\mid\mathbf{s}_i)-\pi(\cdot\mid\mathbf{s}_i),\, r_i(\mathbf{s}_i,\cdot)\rangle \;\gtrsim\; \eta\left(\mathrm{Var}_\pi\big[r_i(\mathbf{s}_i,a_i)\big]+\mathbb{E}_\pi\big[r_i(\mathbf{s}_i,a_i)\,A^\pi(\mathbf{s}_i,a_i)\big]\right), \tag{10}$$

for some $\eta=\Theta(\gamma)$. Combining Eq. 9 and Eq. 10, and weakening $d_\rho^{\pi'}$ to ρ (componentwise monotonicity) gives

$$\mathbb{E}_{\mathbf{s}_i\sim\rho}\big[V^{\pi'}(\mathbf{s}_i)-V^{\pi}(\mathbf{s}_i)\big] \;\gtrsim\; \eta\,\mathbb{E}_{\mathbf{s}_i\sim\rho}\Big[\mathrm{Var}_\pi\big[r_i(\mathbf{s}_i,a_i)\big]+\mathbb{E}_\pi\big[r_i(\mathbf{s}_i,a_i)\,A^\pi(\mathbf{s}_i,a_i)\big]\Big]. \tag{11}$$

Variance decomposition with covariance. Next, using $r_i=r_{i,\text{rea}}+r_{i,\text{tab}}$, we have

$$\mathrm{Var}_\pi\big[r_i(\mathbf{s}_i,a_i)\big]=\mathrm{Var}_\pi\big[r_{i,\text{rea}}(\mathbf{s}_i,a_i)\big]+\mathrm{Var}_\pi\big[r_{i,\text{tab}}(\mathbf{s}_i,a_i)\big]+2\,\mathrm{Cov}_\pi\big(r_{i,\text{rea}}(\mathbf{s}_i,a_i),r_{i,\text{tab}}(\mathbf{s}_i,a_i)\big). \tag{12}$$

Substituting into Eq. 11 complete our proof of Proposition B.3 (Eq. 6). □

Covariance elimination under our reward design. By construction in our setup (see Section 4.1), for each state–action pair $(\mathbf{s}_i,a_i)$, the two components of the PRM signal, i.e., table reward and reasoning reward, are *mutually exclusive*. Formally, we have

$$r_{i,\text{tab}}(\mathbf{s}_i,a_i)\in\{-1,0,1\},\quad r_{i,\text{rea}}(\mathbf{s}_i,a_i)\in\{-1,0,1\},\quad \text{and}\quad r_{i,\text{tab}}(\mathbf{s}_i,a_i)\,r_{i,\text{rea}}(\mathbf{s}_i,a_i)=0.$$

Policy-gradient updates are invariant to adding any per-state baseline, so we may center each component without loss, i.e.,

$$\tilde r_{i,\text{rea}}(\mathbf{s}_i,a_i)\;=\;r_{i,\text{rea}}(\mathbf{s}_i,a_i)-\mathbb{E}_\pi\big[r_{i,\text{rea}}(\mathbf{s}_i,a_i)\big],\qquad \tilde r_{i,\text{tab}}(\mathbf{s}_i,a_i)\;=\;r_{i,\text{tab}}(\mathbf{s}_i,a_i)-\mathbb{E}_\pi\big[r_{i,\text{tab}}(\mathbf{s}_i,a_i)\big].$$

Mutual exclusivity yields $\mathbb{E}_\pi\big[\tilde r_{i,\text{rea}}(\mathbf{s}_i,a_i)\,\tilde r_{i,\text{tab}}(\mathbf{s}_i,a_i)\big]=0$, hence $\mathrm{Cov}_\pi\big(\tilde r_{i,\text{rea}},\tilde r_{i,\text{tab}}\big)=0$ and

$$\mathrm{Var}_\pi\big[\tilde r_i(\mathbf{s}_i,a_i)\big]=\mathrm{Var}_\pi\big[\tilde r_{i,\text{rea}}(\mathbf{s}_i,a_i)\big]+\mathrm{Var}_\pi\big[\tilde r_{i,\text{tab}}(\mathbf{s}_i,a_i)\big],\quad \tilde r_i\;\triangleq\;\tilde r_{i,\text{rea}}+\tilde r_{i,\text{tab}}.$$

Plugging these centered quantities into the bounds of Proposition B.3 (which is NPG-invariant under per-state centering) gives exactly Theorem 4.1's inequality:

$$\begin{aligned}\mathbb{E}_{\mathbf{s}_i\sim\rho}\Big[V^{\pi'}(\mathbf{s}_i)-V^{\pi}(\mathbf{s}_i)\Big] \;\gtrsim\; \mathbb{E}_{\mathbf{s}_i\sim\rho}\Big[&\mathrm{Var}_\pi\big[r_{i,\text{rea}}(\mathbf{s}_i,a_i)\big]+\mathrm{Var}_\pi\big[r_{i,\text{tab}}(\mathbf{s}_i,a_i)\big]\\ &+\mathbb{E}_\pi\big[r_i(\mathbf{s}_i,a_i)\,A^\pi(\mathbf{s}_i,a_i)\big]\Big],\end{aligned} \tag{13}$$

which completes the proof of Theorem 4.1. □

Remarks. (i) Proposition B.3 is strictly more general; Theorem 4.1 follows as a corollary under mutual exclusivity plus per-state centering (baseline invariance). (ii) Mutual exclusivity alone yields $\mathbb{E}_\pi[r_{i,\text{rea}}\,r_{i,\text{tab}}]=0$, but per-state centering is what ensures $\mathrm{Cov}_\pi(r_{i,\text{rea}},r_{i,\text{tab}})=0$. (iii) The alignment term necessarily uses the composite signal r_i because the NPG step is guided by $A^\pi+r_i$.

C Error Analysis

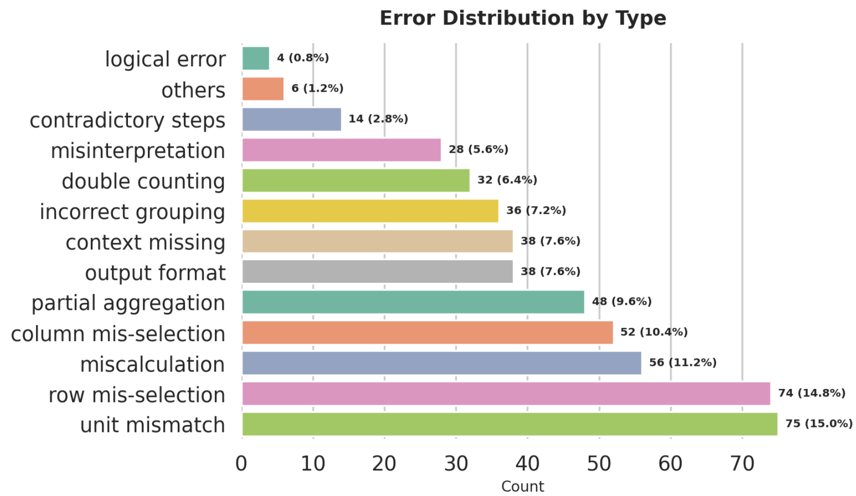

Figure 6: Error distribution over 500 incorrect LRM responses after Best-of-N. The errors are grouped into 13 predefined types, with the majority arising from table retrieval and schema interaction.

In Section 3, we perform a fine-grained error analysis on 500 erroneous responses sampled after Best-of-N selection with Qwen2.5-Math-PRM-72B, to better understand the limitations of LRMs and PRMs. Each response is inspected and categorized by human experts into 13 predefined error types, covering both reasoning and table-specific mistakes. Figure 6 illustrates the overall error distribution.

Error Type Distribution. The most frequent errors are *unit mismatch* (15.0%), *row mis-selection* (14.8%), and *miscalculation* (11.2%). Other common issues include *column mis-selection* (10.4%), *partial aggregation* (9.6%), and missing or incomplete *context* (7.6%). Less frequent but still notable categories include *output format errors*, *incorrect grouping*, *double counting*, *misinterpretation*, and *contradictory steps*. A small portion of errors is grouped under *others* and *logical errors*. This diverse distribution highlights that model failures are not restricted to arithmetic slips but extend to schema understanding and structural reasoning.

Mapping to Reasoning-Step Categories. To reveal deeper patterns, we align the 13 error types with four reasoning-step categories reflecting the typical flow of LRMs:

- **Table Retrieval Step**: Includes row/column mis-selection, unit mismatch, and partial aggregation. These account for 47.7% of total errors, indicating difficulty in locating and extracting the correct table region.
- **Schema Interaction Step**: Covers miscalculation, grouping mistakes, double counting, and misinterpretation of table semantics. This represents 34.3% of errors, reflecting challenges in reasoning over structured contents once retrieved.
- **Inner-Thinking Step**: Logical errors or contradictory reasoning steps independent of table contents. These contribute 12.0% of total errors, suggesting LRMs remain relatively competent in pure logical chains compared to table-centric operations.
- **Others**: Errors arising from context omission or improper output formatting.

Key Findings. The analysis confirms that most model weaknesses lie in table-related operations, including table retrieval and schema interaction, rather than general logical reasoning. PRMs, when supervising such steps, face greater challenges since they must not only validate the correctness of reasoning but also verify alignment between the retrieved sub-table and the query.

D Experimental Setups

D.1 Policy Model Configurations

In our experiments, we adopt an LRM DeepSeek-R1-Distill-Qwen-14B [13] as the downstream policy model. During inference, we configure the model with a temperature of 0.7, a maximum generation length of 16,384 tokens, and top-p sampling with $p = 0.95$. We evaluate the LRM on several inference-time scaling strategies:

Best-of-N (BoN). The policy model generates N candidate responses independently. A verifier (PRM) scores each response, and the final output is selected based on a voting or scoring method.

Beam Search. Given beam width N and branching factor M, the model generates N initial steps. The verifier then selects the top N/M continuations, and the model expands each with M new candidates. This process repeats until termination, enabling guided exploration of high-quality reasoning paths.

Diverse Verifier Tree Search (DVTS). DVTS is a variant of beam search where the search process is divided into multiple subtrees. Each subtree is explored independently using verifier-guided expansions, with candidates selected at every step based on PRM scores.

Majority Voting. After generating multiple responses, the final answer is determined by simple majority over identical outputs, regardless of intermediate step scores. This method provides a baseline aggregation mechanism.

LLM-as-a-Judge. Instead of relying solely on PRMs, a separate LLM is prompted to compare and evaluate candidate responses directly, selecting the most plausible or logically consistent output.

D.2 Training Details

We train TATTO using the off-the-shelf Qwen-3-8B model [61] on our curated 60K dataset. For supervised fine-tuning, we adopt the LLaMA-Factory framework [76]. The training setup uses a learning rate of 1×10^{-5}, a weight decay of 1×10^{-4}, a maximum sequence length of 20,000, and is run for 3 epochs. For the RL training stage, we adopt the VeRL framework [45] to further optimize the SFT checkpoint via policy optimization. The model is trained with a batch size of 32, generating 8 samples per question as the group size, and is run for 3 epochs. During inference, we use the OpenR framework [53] to deploy our trained TATTO-8B, which serves as a verifier to guide the downstream LRM under different test-time scaling strategies.

E Ablations

Table 4: Ablation on confidence calibration λ_{cal}.

N=32	TB-NR	TB-FC	TB-DA
0.3	76.8	80.9	33.1
0.5	77.3	81.3	33.6
0.8	78.1	**82.0**	**34.3**
1.0	**78.5**	81.4	33.8

Table 5: Ablation on tool-grounding λ_{tool}.

N=32	TB-NR	TB-FC	TB-DA
0.1	75.2	76.3	30.8
0.5	75.9	76.9	32.2
1.0	**78.1**	**82.0**	34.3
1.3	77.5	81.2	**34.6**

Ablations on λ_{cal} and λ_{tool}. In Eq. 2, we use λ_{cal} and λ_{tool} as tunable coefficients to balance the contributions of the corresponding reward terms in GRPO. To examine their influence, we separately train our verifier model (initialized from the same SFT checkpoint) by varying $\lambda_{\text{cal}} \in \{0.3, 0.5, 0.8, 1.0\}$ and $\lambda_{\text{tool}} \in \{0.1, 0.5, 1.0, 1.5\}$ during RL, and then evaluate on TableBench with N = 32. As shown in Table 4 and 5, performance improves as λ_{cal} increases, peaking at 0.8–1.0. For λ_{tool}, accuracy rises steadily and is strongest around 1.0–1.3. These results empirically confirm the effectiveness of confidence calibration and tool-grounding in enhancing TTS.

F Case Study on TATTO

Before RL Training

Okay, I need to verify step by step for the provided answer and determine if each step is correct or not.

[Step 0 Evaluation]
For step 0, let me first check whether it retrieves the correct information from the input table …
…

[Step 3 Evaluation]
Looking at the step 3 and the table data provided, I need to verify whether the summarization calculation on the column "capacity in persons / hour\" is correct. Let me add each part together:

350 + 500 + 970 + 10000 +
2000 + 2800 + 650 + 650 +
1000 + 1000 + 720 + 820 + 1000 = 16,920 *--- Wrong Calculation*

This is not equal to the result in the providing answer… making the step incorrect.

The step is \\boxed{incorrect}

After RL Training

Alright, I am given a complete table, question, and a corresponding answer. I need to check the correctness on each step of the answer. Let me verify step by step:

[Step 0 Evaluation]
I start by checking if Step 0 pulls the proper information from the table…
…
[Step 3 Evaluation]
To verify Step 3, I need to check the summarization (sum) over the column "capacity in persons / hour".
<code>
```python
table = {
    "columns": ["name or route", "capacity in persons / hour"],
    "data": [
        ["betten betten village", 350],
        ["betten village bettmeralp", 500],
        ...
        ["trainerlift", 820],
        ["lager 1", 1000],
    ],
}
capacity_index = table["columns"].index("capacity in persons / hour")
total_capacity = sum(row[capacity_index] for row in table["data"])

print("Total capacity in persons/hour:", total_capacity)
```
</code>
<interpreter>Total capacity in persons/hour: 22460 </interpreter>
Since the computed total of 22,460 aligns with the result in step 3, this step is correct.

The step is \\boxed{correct}

Figure 7: Case Study on TATTO Before and After RL Training. Applying RL training enhances our PRM's dynamic utilization of tool calls, which in turn provides more reliable supervision over the input reasoning trajectories of LRMs.

G Limitations and Broader Impacts

While reinforcement learning with reward shaping enhances our PRM's ability to capture fine-grained tabular reasoning signals, it introduces more computational overhead. Compared to SFT-only training, the RL stage requires additional rollouts, reward evaluations, and optimization steps, which can increase training cost and resource demands. This overhead may hinder reproducibility and accessibility in low-resource environments, motivating future work on more efficient reward objectives and lightweight reward modeling strategies. In addition, our current framework is limited to text–table reasoning, and extending it to multimodal settings (e.g., integrating charts or image-based tables) remains an important direction for future work.

Broader Impacts. From a broader perspective, this work highlights the potential for process reward models to enhance structured reasoning in domains such as fact-checking, scientific analysis, and decision support. At the same time, reliance on automated verification carries risks: if tools or training data contain errors, these may be amplified rather than corrected. We encourage future research to explore mechanisms for auditing verifier reliability, reducing the energy footprint of RL training, and ensuring equitable performance across diverse application domains.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our claims are summarized properly in our Introduction section.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitation discussion is included in the Appendix.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide detailed assumptions and complete proof in the Appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides detailed descriptions of the method implementation, training details, and experimental setup in the Experiment Section and Appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide detailed implementation frameworks and dataset instructions in the Appendix.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Experiment details are provided in detail in the Appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Each experiment's results are reported as the average over three independent experimental runs.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The experimental details are provided in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We comply with the Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We include the broader impact discussion in the Appendix.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not observe such risks of misuse in this paper.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All existing resources are properly cited in this paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: This paper does not release any real-world dataset. Other code implementations are detailed as supplementary material.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No human objects are involved in this paper.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The LLM usage is described in detail in this paper.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.