CSPO: Alleviating Reward Ambiguity for Structured Table-to-LaTeX Generation

Yunfan Yang[1]*, **Cuiling Lan**[2]†, **Jitao Sang**[1,3]†, **Yan Lu**[2]
[1]Beijing Key Laboratory of Traffic Data Mining and Embodied Intelligence, Beijing Jiaotong University,
[2]Microsoft Research Asia, [3]Peng Cheng Laboratory
{yunfanyang, jtsang}@bjtu.edu.cn, {culan, yanlu}@microsoft.com

Abstract

Tables contain rich structured information, yet when stored as images their contents remain "locked" within pixels. Converting table images into LaTeX code enables faithful digitization and reuse, but current multimodal large language models (MLLMs) often fail to preserve structural, style, or content fidelity. Conventional post-training with reinforcement learning (RL) typically relies on a single aggregated reward, leading to reward ambiguity that conflates multiple behavioral aspects and hinders effective optimization. We propose **C**omponent-**S**pecific **P**olicy **O**ptimization (CSPO), an RL framework that disentangles optimization across LaTeX tables components—structure, style, and content. In particular, CSPO assigns component-specific rewards and backpropagates each signal only through the tokens relevant to its component, alleviating reward ambiguity and enabling targeted component-wise optimization. To comprehensively assess performance, we introduce a set of hierarchical evaluation metrics. Extensive experiments demonstrate the effectiveness of CSPO, underscoring the importance of component-specific optimization for reliable structured generation.

1 Introduction

Scientific documents often contain complex tables that encapsulate critical data and insights (Gemelli et al., 2024; Xia et al., 2024; Jiang et al., 2025). However, when these tables are embedded in images—such as screenshots or PDFs—their structured information becomes locked within pixels, hindering data extraction, analysis, and reuse. Accurately converting table images into structured LaTeX code is therefore a crucial step toward reliable digitalization and seamless editing.

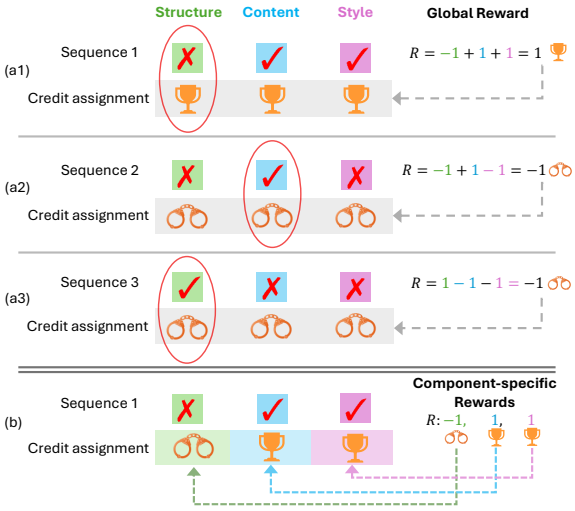

Figure 1: Motivating example of reward ambiguity in table image-to-LaTeX generation. 🏆 denotes a positive reward, while 👓 denotes a penalty. (a) Given a table image, multiple LaTeX sequences are generated with varied errors in structure, content, and style (errors marked with ✗). Using a single global reward leads to **reward ambiguity**, where (a1) an incorrect *structure* is erroneously reinforced, (a2) correct *content* is unfairly penalized, and (a2) and (a3) receive identical rewards despite differing component fidelity. (b) Our method mitigates this by decomposing the LaTeX code into functional components and **assigning component-specific rewards for targeted optimization, thereby alleviating reward ambiguity**.

Recent studies have explored the vision-to-code problem through specialized systems tailored for table understanding. LATTE (Jiang et al., 2025) introduces iterative error localization and correction, while DocGenome (Xia et al., 2024) fine-tunes Pix2Struct (Lee et al., 2023) for table parsing. Beyond these task-specific efforts, multimodal large language models (MLLMs) such as GPT-4o and Qwen2.5-VL have demonstrated strong visual-to-text generalization, enabling zero-shot LaTeX generation. Nevertheless, both specialized systems and general MLLMs often introduce structural incon-

* Work done during internship at Microsoft Research Asia.
† Corresponding authors.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 25374–25391

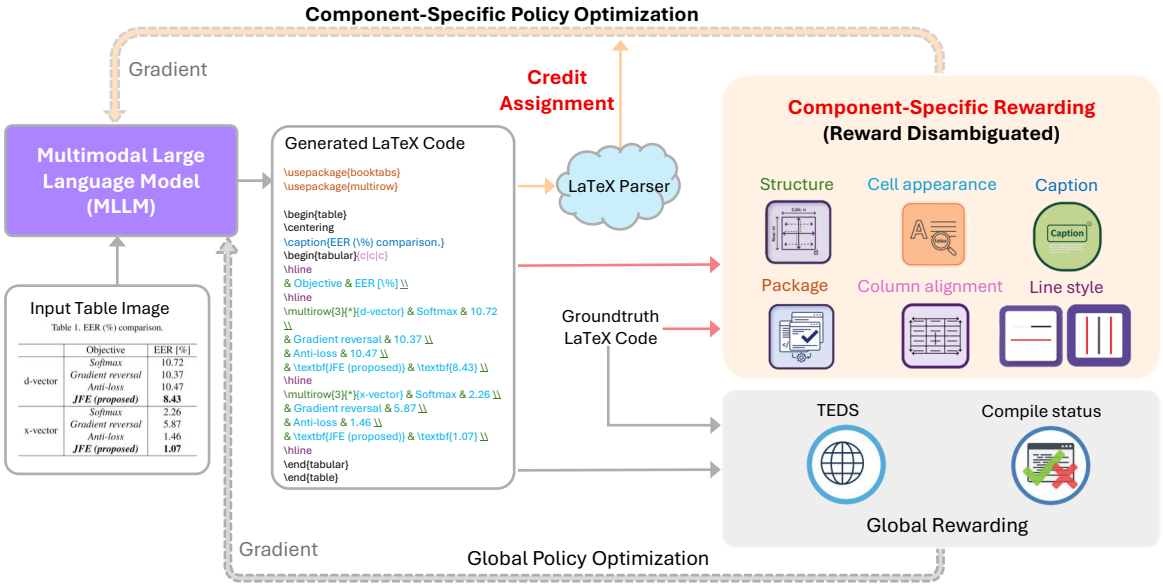

Figure 2: Overview of **Component-Specific Policy Optimization (CSPO)**. Particularly, CSPO decomposes each generated code sequence into functional components (*e.g.*, structure, cell appearance, caption, package inclusion, alignment, and line style) using a LaTeX parser. It conducts **component-specific rewarding** by assessing each component's fidelity (with a strong LLM as the judge), performs **component-specific credit assignment** and **optimization**, effectively mitigating reward ambiguity in table image-to-LaTeX generation.

sistencies (*e.g.*, incorrect cell merges), lose fine-grained formatting details (*e.g.*, mismatched lines), content mistakes, or generate non-compilable code. This motivates the question: how can we effectively align MLLMs for this highly structured table generation?

Reinforcement learning (RL) has become a dominant paradigm for post-training alignment, yielding substantial improvements in reasoning, programming, and mathematical problem solving (Ouyang et al., 2022; Qu et al., 2025; Yu et al., 2025b; Guo et al., 2025). However, its use in table image-to-LaTeX generation remains largely underexplored. Unlike free-form language tasks (*e.g.*, visual question answering), table-to-LaTeX generation presents distinct challenges, including multi-faceted fidelity (covering structure, content, style), hierarchical syntax (*e.g.*, properly nested tabular and multicolumn structures), and execution sensitivity (the code part in Figure 2 shows an example of LaTeX code sequence).

Existing RL approaches typically compute a single aggregated reward across the entire output sequence (Shao et al., 2024; Yu et al., 2025a; Ling et al., 2025). For such a highly structured sequence generation task, this global aggregation is problematic, as it introduces **reward ambiguity—where fundamentally heterogeneous aspects such as structure, content, and style are collapsed into a single undifferentiated signal/reward.** Consequently, the model struggles to assign credit accurately, **leading to unreliable gradients and limited fidelity improvements**. Figure 1 illustrates this issue: in (a1), an incorrect structure component receives a positive global reward, mistakenly reinforcing the error; in (a2), the correct content component is wrongly penalized; (a2) and (a3) receive identical aggregated rewards despite differing in component fidelity, failing to distinguish good from bad performance in individual components.

To address this challenge, we propose Component-Specific Policy Optimization (CSPO), an RL framework specifically designed to mitigate reward ambiguity by assigning dedicated rewards to distinct functional components of LaTeX tables. As illustrated in Figure 2, CSPO combines a global reward that captures overall output quality with component-specific rewards that disentangle structure, content, and style. During RL training, CSPO employs a LaTeX parser to decompose each generated code sequence into fine-grained functional components, including package imports, structure, cell appearance, captions, alignment, and line style. It then performs **component-specific rewarding** by evaluating each component's fidelity, conducts **component-specific credit**

assignment and optimization, leading to more reliable component-level policy optimization. This approach ensures that improvements in one component are not overshadowed by errors in others, thereby enhancing the model's ability to generate faithful LaTeX code.

Furthermore, to enable more diagnostic and interpretable evaluation, we introduce a suite of **hierarchical metrics**. Beyond global similarity and compilation checks, these metrics separately measure structural correctness (*e.g.*, row/column spans), content fidelity (*e.g.*, cell values), and stylistic consistency (*e.g.*, line style, font styles), providing granular feedback on model performance.

Our contributions are summarized as follows:

- We identify **reward ambiguity** as a fundamental challenge in RL-based structured sequence generation for table-to-LaTeX conversion.
- We propose CSPO, an effective RL framework that addresses reward ambiguity through **component-specific rewarding** and **explicit credit assignment**, enabling reliable and controllable **component-specific optimization**.
- We introduce **hierarchical evaluation metrics** for comprehensive and diagnostic assessment, providing useful signals for guiding future table-to-LaTeX model design and optimization.

Experimental results demonstrate the effectiveness of our CSPO, highlighting the importance of addressing reward ambiguity and offering a general blueprint for structured sequence generation tasks.

2 Related Work

Table Image to Structured Markup. Research on image-based table recognition has evolved from early detection and structure-parsing pipelines to end-to-end systems that directly map images to structured markup. Benchmarks such as PubTabNet (Zhong et al., 2020) were pivotal in this transition, introducing large-scale supervision for image-to-HTML conversion. Encoder–decoder architectures (*e.g.*, EDD Zhong et al., 2020) focused on HTML/XML outputs, motivating subsequent methods in image-to-structure generation.

Some recent studies adopted LaTeX as the target format for its precise layout control, publication-quality rendering, and seamless integration with scientific workflows—capabilities that HTML lacks. DocGenome (Xia et al., 2024) fine-tuned Pix2Struct (Lee et al., 2023) for table image-to-LaTeX conversion. LATTE (Jiang et al., 2025) introduced iterative refinement with delta-view correction to improve renderable LaTeX extraction from PDFs. Concurrent with our work, Ling et al. (2025) post-train MLLMs with RL, using a single aggregated global reward signal. Despite these advances, these models still struggle to faithfully reconstruct tables, often producing misaligned structures, inconsistent formatting, or incorrect cell content. In this work, we identify reward ambiguity as a key challenge limiting the effectiveness of RL-based post-training and propose CSPO to address it.

Post-training Alignment and RL. LLMs and MLLMs have advanced rapidly, showing strong generalization across diverse tasks. To further enhance domain-specific skills, RL-based post-training has become a widely adopted strategy for alignment and performance improvement (Ouyang et al., 2022; Qu et al., 2025; Yu et al., 2025b; Guo et al., 2025; Perera et al., 2025; Lai et al., 2025; Tang et al., 2025; Jia et al., 2025). Methods such as GRPO (Shao et al., 2024) have shown effectiveness in mathematical reasoning (Yu et al., 2025a), program synthesis (Tang et al., 2025), and multimodal analysis (Zhou et al., 2025; Lai et al., 2025).

However, most approaches rely on a single aggregated reward that holistically evaluates outputs (Shao et al., 2024; Ling et al., 2025), leading to reward ambiguity and unreliable optimization when applied to table image-to-LaTeX generation. We address this limitation through disambiguated credit assignment for component-specific policy optimization.

3 Problem Formulation

We study the task of **table image-to-LaTeX generation**, which aims to generate a compilable LaTeX code that faithfully reconstructs a given table image in terms of structure, content, and style. Given an input table image $\mathbf{x}$ and a LaTeX sequence $\mathbf{y} = (y_1, \ldots, y_T)$ of length T, a policy model π_θ defines a conditional distribution:

$$\pi_\theta(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T} \pi_\theta(y_t \mid y_{<t}, \mathbf{x}). \tag{1}$$

To optimize generation quality, a natural approach is post-training the policy model with RL, which maximizes the expected reward:

$$\mathcal{J}(\theta) = \mathbb{E}_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}\big[R(\mathbf{y}, \mathbf{y}^*)\big], \tag{2}$$

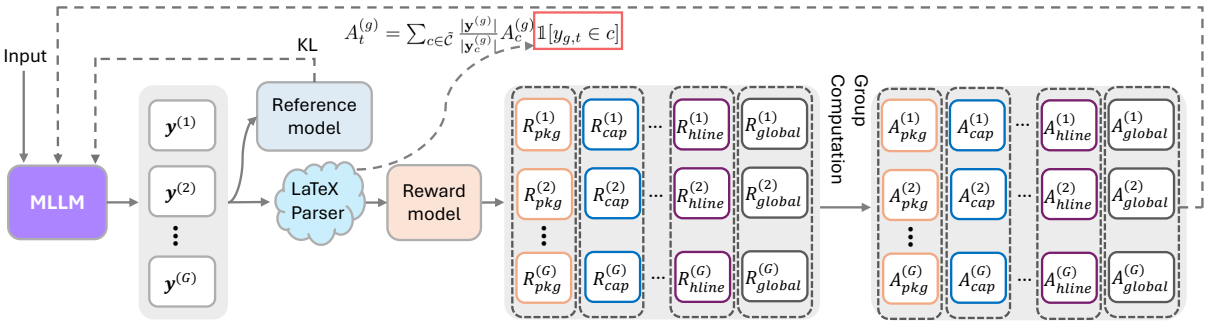

Figure 3: Illustration of proposed CSPO algorithm. Each component-specific advantage $A_c^{(g)}$ ($c \in \tilde{\mathcal{C}}$) is calculated, and credited exclusively to its relevant tokens in rollout g.

where $R(\mathbf{y}, \mathbf{y}^*)$ measures the consistency to the reference $\mathbf{y}^*$, *e.g.*, via Tree-Edit-Distance-based Similarity (TEDS) metric (see Appendix A).

4 Component-Specific Policy Optimization

Reward Ambiguity. However, such a single holistic reward R (*e.g.*, TEDS) conflates heterogeneous aspects of model behavior—structure, content, and style— making it difficult to discern which parts of the output are correct or erroneous (see Figure 1). This reward ambiguity motivates our component-specific approach introduced next.

We propose Component-Specific Policy Optimization (CSPO), with the overall pipeline shown in Figure 2. In particular, CSPO performs component-specific policy optimization through component decomposition, component-specific rewarding, credit assignment and optimization, enabling reliable credit attribution and policy updates for faithful LaTeX code generation.

4.1 Component Decomposition

LaTeX tables exhibit multi-dimensional fidelity, where correctness depends jointly on the *structural layout* (*e.g.*, rows, columns, merged cells), *contents* (*e.g.*, cell text and numbers), *stylistic attributes* (*e.g.*, alignment, line styles, boldface), and *compilability*.

To facilitate disambiguated rewarding and credit assignment, we build a rule-based LaTeX parser to decompose each LaTeX sequence into seven functional components:

$$\mathcal{C} = \{\text{pkg}, \text{cap}, \text{struct}, \text{cell-app}, \text{align}, \text{vline}, \text{hline}\}, \tag{3}$$

which includes package dependencies (pkg), caption correctness (cap), structural organization (struct), cell appearance (cell-app), column alignment (align), and rule placement (vline, hline). Please see Appendix B for more details. We illustrate the decomposition by marking different components with different colours over the generated LaTeX code in Figure 2.

4.2 Component-Specific Rewarding

We assign dedicated rewards to each functional component of a LaTeX sequence to achieve disambiguated rewarding. Formally, for a generated sequence $\mathbf{y}$ and its reference $\mathbf{y}^*$, we denote the component-specific rewards as

$$\mathcal{R}(\mathbf{y}, \mathbf{y}^*) = \{R_c \mid c \in \mathcal{C}\}, \tag{4}$$

where R_c measures fidelity of component c. A strong LLM (*e.g.*, GPT-4o) serves as an automatic judger (see Appendix C for prompt) to evaluate each component. Each component reward is binary—1 if consistent with reference, 0 otherwise—providing clear, localized signals that disambiguate rewards across components.

4.3 Credit Assignment and Optimization

With each functional component in $\mathcal{C}$ assigned a dedicated reward, we attribute credit only to the tokens corresponding to that component, avoiding cross-component interference.

As illustrated in Figure 3, CSPO extends Group Relative Policy Optimization (GRPO) by augmenting the global reward with component-specific rewards. Given a group of rollouts $\{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(G)}\}$ sampled from the MLLM policy π_θ, the **component-specific advantage** for component c in rollout g is defined as:

$$A_c^{(g)} = \frac{R_c^{(g)} - \mu_c}{\sigma_c + \epsilon}, \tag{5}$$

where $R_c^{(g)}$ represents the reward of component c in rollout g, and μ_c and σ_c denote the mean and standard deviation of the rewards for this component, respectively.

For the t-th token, we define the **token-level component advantage** w.r.t. component c as

$$A_{c,t}^{(g)} = A_c^{(g)} \cdot \mathbb{I}[y_t^{(g)} \in c], \tag{6}$$

where $\mathbb{1}[\cdot]$ is an indicator function that activates only when the generated token $y_t^{(g)}$ belongs to component c. **This masking mechanism ensures that gradient updates are applied exclusively to relevant tokens, thereby facilitating precise component-specific optimization.**

While $R_c^{(g)}$ optimizes component-wise fidelity, it may overlook inter-component dependencies. To compensate, we incorporate two global signals: the *TEDS score* R_{TEDS} for overall similarity, and a *compile reward* R_{cmp} to penalize non-compilable outputs. Their sum forms the global reward $R_{global}^{(g)} = R_{\text{TEDS}}^{(g)} + R_{\text{cmp}}^{(g)}$, whose normalized advantage is shared across all tokens within rollout g as $A_{global,t}^{(g)} = A_{global}^{(g)}$. For unified formulation, we extend the component set in (3) to include this global component as:

$$\tilde{\mathcal{C}} = \mathcal{C} \cup \{\text{global}\}. \tag{7}$$

The CSPO objective maximizes the expected normalized advantage while regularizing the policy via a KL penalty to the reference model:

$$\mathcal{J}_{\text{CSPO}}(\theta) = \mathbb{E}_{(x,\mathbf{y}^*)\sim\mathcal{D},\{\mathbf{y}^{(g)}\}_{g=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\Big[\mathcal{L}(\theta) - \beta D_{\text{KL}}(\pi_\theta||\pi_{\text{ref}})\Big], \text{ where } \mathcal{L}(\theta) = \sum_{c\in\tilde{\mathcal{C}}} \mathcal{L}_c(\theta). \tag{8}$$

Each component-specific objective $\mathcal{L}_c(\theta)$ adopts a GRPO-style clipped surrogate loss:

$$\mathcal{L}_c(\theta) = \frac{1}{G}\sum_{g=1}^{G}\frac{1}{|\mathbf{y}_c^{(g)}|}\sum_{t=1}^{|\mathbf{y}_c^{(g)}|}\min\Big[\rho_{g,t}(\theta)A_{c,t}^{(g)}, \text{clip}(\rho_{g,t}(\theta), 1-\epsilon, 1+\epsilon)A_{c,t}^{(g)}\Big], \tag{9}$$

where $|\mathbf{y}_c^{(g)}|$ denotes the number of tokens associated to component c, and ϵ is the clipping threshold.

Based on (8)(9), the overall objective $\mathcal{L}(\theta)$ can be reformulated (derivation in Appendix D) as

$$\mathcal{L}(\theta) = \frac{1}{G}\sum_{g=1}^{G}\frac{1}{|\mathbf{y}^{(g)}|}\sum_{t=1}^{|\mathbf{y}^{(g)}|}\min\Big[\rho_{g,t}(\theta)A_t^{(g)}, \text{clip}(\rho_{g,t}(\theta), 1-\epsilon, 1+\epsilon)A_t^{(g)}\Big], \tag{10}$$

Algorithm 1 Component-Specific Policy Optimization (CSPO)

Require: Pretrained policy π_θ, reference policy π_{ref}, dataset $\mathcal{D}$, group size G, w_c

1: **for** each training iteration **do**
2: Sample a batch of table images $x \sim \mathcal{D}$
3: Generate G rollouts $\{\mathbf{y}^{(g)}\}_{g=1}^{G} \sim \pi_\theta(\cdot|x)$
4: Decompose each $\mathbf{y}^{(g)}$ into components $c \in \mathcal{C}$
5: Compute component rewards $R_c^{(g)}$ and normalize: $A_c^{(g)} = (R_c^{(g)} - \mu_c)/(\sigma_c + \epsilon)$, where $\epsilon = 1 \times 10^{-4}$
6: Aggregate token-level advantages: $A_t^{(g)} = \sum_{c\in\tilde{\mathcal{C}}} \frac{|\mathbf{y}^{(g)}|}{|\mathbf{y}_c^{(g)}|} w_c A_c^{(g)} \mathbb{1}[y_t^{(g)} \in c]$
7: Update π_θ by maximizing the clipped CSPO objective in (8).
8: **end for**

where $A_t^{(g)}$ denotes the **token-level aggregated advantage**, which integrates the contributions of all components:

$$\begin{aligned} A_t^{(g)} &= \sum_{c\in\tilde{\mathcal{C}}} \frac{|\mathbf{y}^{(g)}|}{|\mathbf{y}_c^{(g)}|} A_{c,t}^{(g)} \\ &= \sum_{c\in\tilde{\mathcal{C}}} \frac{|\mathbf{y}^{(g)}|}{|\mathbf{y}_c^{(g)}|} A_c^{(g)} \cdot \mathbb{I}[y_t^{(g)} \in c]. \end{aligned} \tag{11}$$

Here, $|\mathbf{y}^{(g)}|$ is the total token count of rollout g. For the global component, $|\mathbf{y}_{global}^{(g)}| = |\mathbf{y}^{(g)}|$, whereas for others $|\mathbf{y}_c^{(g)}| < |\mathbf{y}^{(g)}|$.

Notation summaries are in Appendix E. Algorithm 1 summarizes the CSPO process. w_c denotes weights for balancing the contributions of different components (see ablation in Appendix F.1).

5 Evaluation Metrics

To systematically assess model capabilities, we introduce a hierarchical evaluation framework that combines global similarity metrics—TEDS for overall matching and compile success rate—with newly proposed fine-grained diagnostics that disentangle structure, style, and content fidelity, enabling more interpretable analysis of model behavior beyond aggregated scores.

- **Tree Edit Distance Similarity** $TEDS$: measures the overall semantic and structural similarity between generated and reference LaTeX renderings (see Appendix A).

- **Compilation Rate** R: the percentage of generated LaTeX codes that compile successfully.

Fine-Grained Metrics. To provide a more granular evaluation, we introduce fine-grained metrics automatically computed by an LLM (*e.g.*, GPT-4o) that compares the predicted code $\mathbf{y}$ with the reference $\mathbf{y}^*$ (see Appendix C for the detailed prompt). Each metric is defined at the *table level*: a score of 1 indicates correctness, and 0 indicates at least one error. We evaluate along three main dimensions:

- **Structural Correctness** (S): Verifies the consistency of structural elements, including merged cells (`\multicolumn`, `\multirow`) and cell positions.
- **Content Fidelity** (C): Checks equivalence of textual and numeric entries with the ground truth.
- **Stylistic Fidelity** (Y): Assesses presentation consistency, including line style (Y_{line}), column/cell alignment (Y_{align})(*e.g.*, left aligned), and cell formatting (Y_{cell})(*e.g.*, boldface, underline), where $Y = Y_{\text{line}} \wedge Y_{\text{align}} \wedge Y_{\text{cell}}$. $\wedge$ denotes logical AND operation.

We further define composite indicators to assess the overall fidelity in terms of structure, content, style, and compilation success:

- **Overall Fidelity** ($OF = S \wedge C \wedge Y \wedge R$): Combined correctness across structure, content, style, and compilation success.

6 Experiments

We conducted extensive experiments to validate the effectiveness of our proposed **CSPO** for table image-to-LaTeX generation.

6.1 Experimental Setup

Dataset. We construct *TableTex*, a benchmark comprising 19000 pairs of table images and renderable LaTeX codes. In order to support complete table generation, each table code contains (i) necessary package declarations to ensure its compilation correctness; (ii) the caption of table[1]; (iii) the body of the table that preserves the rich styles and formatting of the original table. The dataset is curated from scientific articles collected from arXiv, where renderable table code is directly extracted from LaTeX sources. The articles span six major categories—Computer Science, Mathematics, Economics, Electrical Engineering and Systems Science, Quantitative Finance, and Statistics—covering publications from 2012 to 2025. Each extracted LaTeX snippet is rendered into an image, resulting in tables with diverse aspect ratios, resolutions, and visual layouts.

We split the dataset to training set *TableTex-train* of 15000 samples and testing set *TableTex-test* of 4000 samples by a random partition.

We evaluate our method on three benchmarks: one **in-domain dataset**, *TableTex-test*, and two **out-of-domain datasets**, *i.e.*, *DocGenome-table-1k* (Xia et al., 2024), and *Table2LaTeX-test-simple* (Ling et al., 2025). (i) *DocGenome-table-1k* (Xia et al., 2024) is a 1,000-sample subset of DocGenome, where table images are automatically annotated by DocParser and cropped directly from raw PDFs. As a result, the dataset exhibits substantial variability in table localization accuracy, as well as occasional background clutter and partial table truncation, making it a significantly more challenging benchmark for robust table code generation. (ii) *Table2LaTeX-test-simple* consists of 496 test samples from (Ling et al., 2025), where the table captions and colors are excluded during their dataset construction.

Evaluation Metrics. Metrics defined in Section 5 are used for evaluation. By default, we evaluate the models on the fine-grained metrics by using GPT-4o as the judge. Consistent trends were observed when using other LLM judges (see Section 6.4).

Implementation Details. We adopt vision–language models based on multimodal LLM backbones (*i.e.*, Qwen2.5-VL-3B, Qwen2.5-VL-7B) as base models, and train them using a two-stage procedure: supervised fine-tuning (SFT) followed by reinforcement learning (RL). For SFT, we train on 10000 samples for one epoch, with an initial learning rate of 5e-6 and a batch size of 64. For RL, we use 5,000 training samples and train for two epochs, with a rollout batch size of 16 and four gradient accumulation steps. We set the group size G to 8, and the learning rate to 1e-6. We employ a fixed weighting scheme to balance the contributions of different components, *i.e.*, $\mathcal{L}(\theta) = \sum_{c \in \tilde{\mathcal{C}}} w_c \mathcal{L}_c(\theta)$, where we assign $w_{\text{global}} = 3$ to the global component and set $w_c = 1$ for each remaining component. For evaluation, we generate a single rollout for each input image using greedy decoding.

[1]Existing datasets usually lack table captions and sample-wise compilation package specifications, which we include for table completeness.

Model	TEDS ↑	OF ↑	S ↑	C ↑	Y ↑	Y_{line} ↑	Y_{align} ↑	Y_{cell} ↑	R ↑
GPT-4o (OpenAI, 2025a)	72.3	6.8	65.0	73.1	9.4	23.0	54.0	72.5	99.9
Gemini-2.5 Flash (Comanici et al., 2025)	66.2	11.1	78.8	79.2	12.9	33.5	47.2	81.3	99.3
Qwen2.5-VL-72B (Bai et al., 2025)	75.9	10.1	79.6	76.2	12.6	32.1	56.7	69.9	99.7
Nougat (Blecher et al., 2023)	67.9	21.2	68.7	63.3	25.8	37.1	71.9	70.1	99.1
Qwen2.5-VL-3B	66.0	3.1	51.8	59.8	4.5	18.9	34.0	61.7	93.7
Qwen2.5-VL-3B-VSGRPO* (Ling et al., 2025)	87.8	40.4	77.5	86.6	47.6	70.9	71.4	91.9	**99.7**
Qwen2.5-VL-3B-GRPO	87.7	42.0	74.7	86.7	50.3	74.9	71.5	92.0	99.6
Qwen2.5-VL-3B-CSPO (Ours)	**87.9**	**45.2**	**77.6**	**87.0**	**53.6**	**76.5**	**74.2**	**92.1**	99.6
Qwen2.5-VL-7B	64.0	5.3	67.0	63.0	6.4	19.5	46.7	60.7	77.2
Qwen2.5-VL-7B-VSGRPO* (Ling et al., 2025)	89.5	51.1	81.3	89.9	58.2	76.8	79.5	93.9	99.9
Qwen2.5-VL-7B-GRPO	89.7	50.6	80.7	89.7	58.9	**76.8**	80.5	93.7	**99.9**
Qwen2.5-VL-7B-CSPO (Ours)	**89.7**	**53.0**	**81.4**	**90.0**	**60.6**	76.6	**82.7**	**93.9**	99.8

Table 1: Performance comparisons on *TableTex-test* (with fine-grained metrics evaluated by GPT-4o). We report metrics across hierarchical dimensions. **Global metrics**: TEDS, Overall Fidelity (OF), Compilation Rate (R); **Fine-grained metrics**: (i) Structure Fidelity (S), (ii) Content Fidelity (C); (iii) Style Fidelity(Y): Line Style (Y_{line}), Alignment (Y_{align}), Cell Style (Y_{cell}). Higher scores (↑) indicate better performance. Note that all evaluation metrics, initially defined in the $[0, 1]$ range (*e.g.*, scoring and correctness measures), are presented as percentages (%) in all the tables for clarity. * denotes that, for fair comparison, we reimplement the reward design of Ling *et al.* (Ling et al., 2025) within our codebase and dataset.

Method	*DocGenome-table-1k*					*Table2LaTeX-test-simple*				
	TEDS ↑	OF ↑	S ↑	C ↑	Y ↑	TEDS ↑	OF ↑	S ↑	C ↑	Y ↑
GPT-4o (OpenAI, 2025a)	60.2	5.1	55.0	41.2	8.4	70.6	3.4	60.9	66.1	4.6
Gemini-2.5 Flash (Comanici et al., 2025)	60.5	8.0	64.9	57.1	12.5	71.5	10.5	72.0	78.4	12.9
Qwen2.5-VL-72B (Bai et al., 2025)	60.3	5.0	61.2	53.9	8.2	74.7	7.5	64.9	66.5	11.7
Nougat (Blecher et al., 2023)	30.8	0.8	27.7	4.4	8.8	27.0	0.4	20.6	2.8	5.7
Qwen2.5-VL-3B	50.2	1.1	43.5	37.0	1.7	63.5	2.0	39.1	44.2	3.0
Qwen2.5-VL-3B-VSGRPO* (Ling et al., 2025)	71.5	17.8	60.3	53.8	30.9	81.5	22.6	63.3	74.2	30.7
Qwen2.5-VL-3B-GRPO	71.3	18.9	60.0	54.4	33.1	81.5	24.4	58.3	**75.6**	32.1
Qwen2.5-VL-3B-CSPO (Ours)	**72.5**	**21.0**	**63.6**	**54.7**	**34.5**	**82.3**	**26.0**	**64.3**	74.4	**33.9**
Qwen2.5-VL-7B	56.6	3.9	51.2	46.3	6.0	66.4	4.0	54.6	58.7	5.4
Qwen2.5-VL-7B-VSGRPO* (Ling et al., 2025)	73.9	25.9	67.8	63.0	39.6	82.9	33.9	66.9	**80.2**	42.3
Qwen2.5-VL-7B-GRPO	74.7	26.9	68.5	62.7	38.7	83.1	34.7	64.7	79.8	44.8
Qwen2.5-VL-7B-CSPO (Ours)	**74.7**	**29.2**	**68.8**	**63.6**	**41.1**	**83.5**	**37.1**	**69.0**	78.4	**46.0**

Table 2: Generalization performance on *DocGenome-table-1k* and *Table2LaTeX-test-simple* (with fine-grained metrics evaluated by GPT-4o). * denotes that, for fair comparison, we reimplement the reward design of Ling *et al.* (Ling et al., 2025) within our codebase and dataset.

6.2 Quantitative Results

We compare our method with representative baselines: **(i) closed-source multimodal large language models (MLLMs)** (*e.g.*, GPT-4o and Gemini-2.5 Flash); **(ii) open-source MLLMs** (*e.g.*, Qwen2.5-VL-72B, Qwen2.5-VL-3B, Qwen2.5-VL-7B); **(iii) specialized expert model** Nougat (Blecher et al., 2023), which is an open-source system for LaTeX code conversion; **(iv)** for fairness, we compare **baseline models** Qwen2.5-VL-3B/7B-GRPO (trained with SFT, and GRPO (Shao et al., 2024) using only our global reward R_{global}); In addition, we implemented the global reward design from Ling *et al.* (Ling et al., 2025) which aggregates code structure consistency and visual fidelity as a single reward, which we refer to as Qwen2.5-VL-7B-VSGRPO (Ling et al., 2025). Note that these models all suffer from reward ambiguity during RL.

Main Result. Table 1 reports the results on *TableTex-test*. We have five main observations/conclusions. **(i)** Our models Qwen2.5-VL-3B/7B-CSPO achieves the highest overall performance in terms of Overall Fidelity (OF) and TEDS scores, outperforming general-purpose MLLMs and baseline models. **(ii)** Under the same settings, CSPO consistently outperforms GRPO by 3.2%/2.4% on the 3B/7B models in terms of Overall Fidelity (OF), outperforms VSGRPO by 4.8%/1.9% on the 3B/7B models, demonstrating the effectiveness of our component-specific policy

Model	TEDS ↑	OF ↑	S ↑	C ↑	Y ↑
Base	66.0	3.1	51.8	59.8	4.5
SFT	87.1	39.8	75.8	86.1	47.3
GRPO	87.7	42.0	74.7	86.7	50.3
CSPO w/ Comp. Sum	87.7	42.2	76.3	87.0	50.9
CSPO w/o Global	87.7	44.7	**77.9**	86.3	51.2
CSPO (Ours)	**87.9**	**45.2**	77.6	**87.0**	**53.6**

Table 3: Ablation studies on 3B models evaluated on *TableTex-test*.

optimization. **(iii)** CSPO shows consistent improvement across structure (S), content (C), and style fidelity (S), indicating that component-specific optimization effectively alleviates reward ambiguity and drives targeted fidelity enhancement. **(iv)** Qwen2.5-VL-7B-CSPO, with increased model capacity, achieves higher performance than Qwen2.5-VL-3B-CSPO. **(v)** Our fine-grained metrics enable a more diagnostic evaluation than TEDS, which only provides an aggregated score. They reveal that general-purpose MLLMs perform well on structure (S) and content (C) but lag behind on style fidelity (Y), while table-specialized models exhibit more balanced performance. We hope these metrics provide useful signals for guiding future table-to-LaTeX model design and optimization.

Generalization Performance. Table 2 shows the comparisons on two out-of-domain datasets. On DocGenome-table-1k, CSPO achieves the best performance across the 3B and 7B models, delivering consistent gains in structure, content, and style fidelity. Compared to GRPO, Qwen2.5-VL-3B-CSPO and Qwen2.5-VL-7B-CSPO improve Overall Fidelity by 2.1% and 2.3%, respectively. On Table2LaTeX-test-simple, CSPO consistently outperforms GRPO/VSGRPO in terms of Overall Fidelity and TEDS. Overall, the results confirm the effectiveness of CSPO on out-of-domain datasets and model sizes.

6.3 Ablation Studies

Table 3 shows the ablation results to validate the effectiveness of our designs on top of 3B base model (Qwen2.5-VL-3B).

Effect of Reward Ambiguous. Beyond comparing CSPO and GRPO, we further evaluate the variant CSPO w/ Comp. Sum, which naively aggregates the global reward and component-specific rewards into a single scalar reward optimized via GRPO. CSPO outperforms CSPO w/ Comp. Sum by 3% in OF, demonstrating that the performance gain stems from component-specific credit assignment and targeted optimization, rather than merely incorporating additional reward signals.

Effect of Global Rewards. Removing the global reward R_{global} (*i.e.*, CSPO w/o Global), *i.e.*, $w_{global} = 0$ leads to 0.5% performance drops. This highlights the value of incorporating global constraints on overall context.

SFT vs. RL. SFT can effectively warm up the training, which quickly boosts the performance of the base model from 3.1% to 39.8% in terms of OF. RL with GRPO and our CSPO further improve the model capabilities by 2.2% and 5.4%, respectively.

Effect of Specific Components. We study the influence of different component rewards. Table 6 in Appendix F.1 shows that removing structure-related or style-related rewards leads to a significant performance drop, while the impact of content-related reward is comparatively smaller.

Effect of Weight w_c. Appendix F.1 (Table 7) shows the ablation study on component weight w_c.

Effect of Reward Granularity. See Appendix F.1 for the ablation study on the impact of reward granularity.

6.4 Reliability of LLM Evaluation

To ensure the robustness of LLM-based evaluation, we validate it from three aspects.

First, we employ multiple independent LLM judges (*i.e.*, GPT-4o (OpenAI, 2025a), Qwen3-Next-80B (Yang et al., 2025), DeepSeek-v3.2 (Liu et al., 2025), and GPT-5.2 (OpenAI, 2025b)) in testing. Table 4 show that the overall performance trends are largely consistent across different LLM judges for 3B models (see Appendix F.2 for 7B models). This also demonstrates the effectiveness of our method.

Second, we verify strong agreement between LLM judgments and human evaluation (for GPT-4o, approximately 90% consistency on 500 randomly sampled examples).

Third, we repeat the GPT-4o evaluation eight times and observe low variance (0.1–0.4) for metrics. Detailed analyses are provided in Appendix F.2.

6.5 Qualitative Results

We visualize the rendered table images from different models. Figure 4 shows that our CSPO generates correct structure (marked by green box), line style (marked by green arrow), while GRPO suffers on structure (marked by red box) and line style

LLM Evaluators/Judge	Method	Overall Fidelity	Structure Fidelity	Content Fidelity	Style Fidelity
GPT-4o (by default)	SFT	39.8	75.8	86.1	47.3
	GRPO	42.0	74.7	86.7	50.3
	CSPO (Ours)	**45.2**	**77.6**	**87.0**	**53.6**
Qwen3-Next-80b	SFT	37.6	72.0	88.4	47.2
	GRPO	39.7	71.6	88.2	49.9
	CSPO (Ours)	**43.3**	**75.5**	**88.5**	**52.5**
DeepSeek-v3.2	SFT	34.0	75.7	86.5	42.0
	GRPO	35.8	75.8	**87.5**	43.3
	CSPO (Ours)	**39.7**	**78.5**	86.7	**47.3**
GPT-5.2	SFT	30.6	66.1	75.7	38.7
	GRPO	31.1	65.4	75.4	40.2
	CSPO (Ours)	**34.5**	**70.1**	**76.2**	**43.0**

Table 4: Performance comparisons among 3B SFT, GRPO and our CSPO models, with fine-grained metrics evaluated by using different LLM judges.

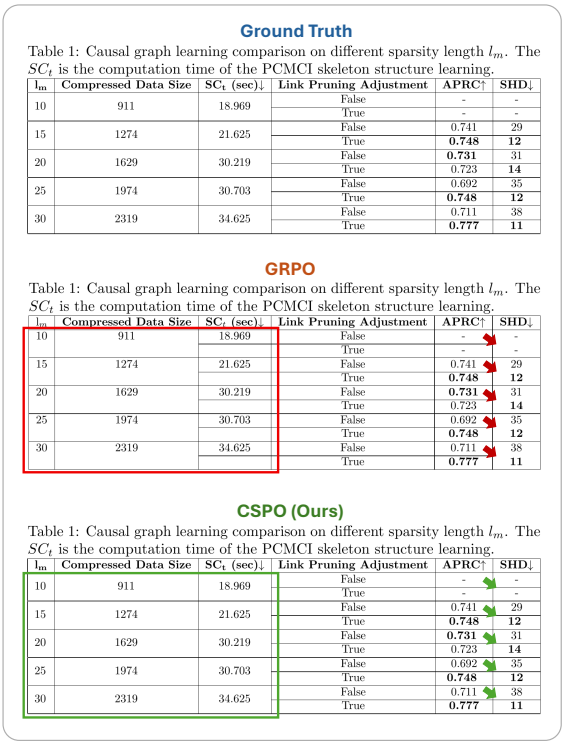

Figure 4: A typical example comparing GRPO and CSPO of 7B models, showing CSPO mitigates **structure** and **line style errors**.

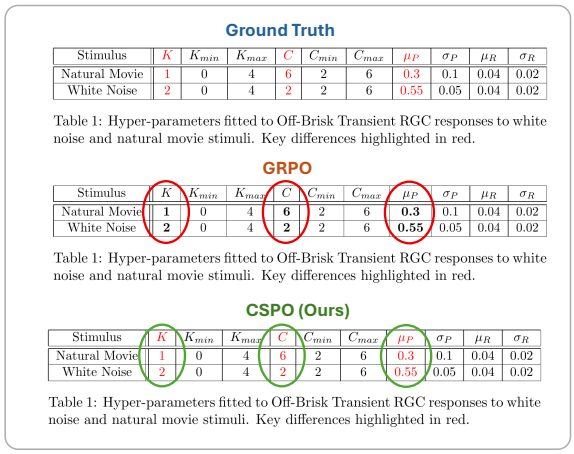

Figure 5: A typical example comparing GRPO and CSPO of 3B models, showing CSPO mitigates **cell style errors**.

(marked by red arrow). Figure 5 shows that CSPO recovers the cell style (marked by ellipse) accurately. See Appendix F.4 for more visualization.

7 Conclusion

We propose Component-Specific Policy Optimization (CSPO), a reinforcement learning framework that alleviates reward ambiguity in table image-to-LaTeX generation through component-specific rewarding, explicit credit assignment, and targeted policy optimization. Extensive experiments on both in-domain and out-of-domain benchmarks demonstrate that CSPO consistently improves structural, style, and content fidelity, highlighting the importance of addressing reward ambiguity in structured sequence generation.

8 Data Consent

We collect data from arXiv, using only papers with CC BY, CC BY-SA, CC0, and CC BY-NC licenses. We will ensure that all collected data is used solely for research purposes, respecting the terms of the respective licenses. No personal or sensitive information is included, and all experiments and model training strictly follow ethical guidelines and data usage policies.

9 Limitations

Our CSPO design has alleviated the reward ambiguous problems and significantly enhanced the performance. However, there are still some limitations. (i) First, the overall fidelity of our models leaves room for further improvement (e.g., 53% for the 7B-CSPO model). (ii) The scale of our training dataset is still small (*i.e.*, 15000 samples). More training data would further improve the performance. (iii) CSPO relies on LLM-based evaluation during training, which introduces additional cost.

References

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. 2023. Nougat: Neural optical understanding for academic documents. *arXiv preprint arXiv:2308.13418*.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Andrea Gemelli, Simone Marinai, Lorenzo Pisaneschi, and Francesco Santoni. 2024. Datasets and annotations for layout analysis of scientific articles. *International Journal on Document Analysis and Recognition (IJDAR)*, 27(4):683–705.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-R1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Caijun Jia, Nan Xu, Jingxuan Wei, Qingli Wang, Lei Wang, Bihui Yu, and Junnan Zhu. 2025. Chartreasoner: Code-driven modality bridging for long-chain reasoning in chart question answering. *arXiv preprint arXiv:2506.10116*.

Nan Jiang, Shanchao Liang, Chengxiao Wang, Jiannan Wang, and Lin Tan. 2025. Latte: Improving latex recognition for tables and formulae with iterative refinement. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 4030–4038.

Yuxiang Lai, Jike Zhong, Ming Li, Shitian Zhao, and Xiaofeng Yang. 2025. Med-R1: Reinforcement learning for generalizable medical reasoning in vision-language models. *arXiv preprint arXiv:2503.13939*.

Kenton Lee, Mandar Joshi, Iulia Raluca Turc, Hexiang Hu, Fangyu Liu, Julian Martin Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. 2023. Pix2struct: Screenshot parsing as pretraining for visual language understanding. In *International Conference on Machine Learning*, pages 18893–18912. PMLR.

Jun Ling, Yao Qi, Tao Huang, Shibo Zhou, Yanqin Huang, Jiang Yang, Ziqi Song, Ying Zhou, Yang Yang, Heng Tao Shen, and 1 others. 2025. Table2latex-rl: High-fidelity latex code generation from table images via reinforced multimodal language models. *arXiv preprint arXiv:2509.17589*.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, and 1 others. 2025. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.

OpenAI. 2025a. Gpt-4o: Gpt-4 with vision capabilities. https://openai.com/research/gpt-4o.

OpenAI. 2025b. Introducing gpt-5.2. https://openai.com/index/introducing-gpt-5-2.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Dilruk Perera, Gousia Habib, Qianyi Xu, Daniel J Tan, Kai He, Erik Cambria, and Mengling Feng. 2025. Beyond prediction: Reinforcement learning as the defining leap in healthcare ai. *arXiv preprint arXiv:2508.21101*.

Xiaoye Qu, Yafu Li, Zhaochen Su, Weigao Sun, Jianhao Yan, Dongrui Liu, Ganqu Cui, Daizong Liu, Shuxian Liang, Junxian He, and 1 others. 2025. A survey of efficient reasoning for large reasoning models: Language, multimodality, and beyond. *arXiv preprint arXiv:2503.21614*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Lingxiao Tang, He Ye, Zhongxin Liu, Xiaoxue Ren, and Lingfeng Bao. 2025. Codereasoner: Enhancing the code reasoning ability with reinforcement learning. *arXiv preprint arXiv:2507.17548*.

Renqiu Xia, Song Mao, Xiangchao Yan, Hongbin Zhou, Bo Zhang, Haoyang Peng, Jiahao Pi, Daocheng Fu, Wenjie Wu, Hancheng Ye, and 1 others. 2024. Docgenome: An open large-scale scientific document benchmark for training and testing multi-modal large language models. *arXiv preprint arXiv:2406.11633*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025a. DAPO: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Tao Yu, Yi-Fan Zhang, Chaoyou Fu, Junkang Wu, Jinda Lu, Kun Wang, Xingyu Lu, Yunhang Shen, Guibin Zhang, Dingjie Song, and 1 others. 2025b. Aligning multimodal llm with human preference: A survey. *arXiv preprint arXiv:2503.14504*.

Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. 2020. Image-based table recognition: data, model, and evaluation. In *European conference on computer vision*, pages 564–580. Springer.

Guanghao Zhou, Panjia Qiu, Cen Chen, Jie Wang, Zheming Yang, Jian Xu, and Minghui Qiu. 2025. Reinforced mllm: A survey on rl-based reasoning in multimodal large language models. *arXiv preprint arXiv:2504.21277*.

A TEDS

Global Similarity Metric TEDS. Inspired by (Zhong et al., 2020), we adopt the Tree-Edit-Distance-based Similarity (TEDS) score to measure the overall similarity. Particularly, we represent the generated LaTeX code and the ground-truth code as rooted tree structures T_{pred} and T_{gt}, respectively. The root has four types of children: *table caption*, *tabular* (which represents the column alignment manners and column lines), *row entity*, and *line entity*. The *tabular* and *row entity* nodes have multiple leaves to elaborate the table details. For example, under each *row entity*, each cell corresponds to a leaf node. We compute a normalized edit distance:

$$\text{TEDS}(T_{\text{pred}}, T_{\text{gt}}) = 1 - \frac{\text{EditDist}(T_{\text{pred}}, T_{\text{gt}})}{\max(|T_{\text{pred}}|,\ |T_{\text{gt}}|)}, \tag{12}$$

where EditDist denotes tree-edit distance, and $|T|$ denotes the number of nodes in T. The cost of insertion, deletion, editing are all 1. Note that we do not take the package headers into consideration in the TEDS measure.

B Component Decomposition

The components in $\mathcal{C}$ are defined as follows:

- **pkg**: imported packages (*e.g.*, `booktabs`, `multirow`);
- **struct**: tabular structure, including row/column merges and overall layout consistency;
- **cap**: table caption consistency;
- **cell-app**: cell-level appearance, covering textual fidelity and formatting consistency (*e.g.*, bold, underline);
- **align**: column alignment type, specifying whether each column is centered, left-, or right-aligned (`c`, `l`, `r`);
- **vline**: vertical rule placement (`|`);
- **hline**: horizontal rule placement (`\hline`, `\cline`).

C Prompts for Rewarding and Evaluation

C.1 Prompt for Component Rewarding

To automatically provide reward for each component, we leverage a strong LLM as the reward model, *i.e.*, an automatic evaluator, to compare the predicted code $\mathbf{y}$ against the ground-truth $\mathbf{y}^*$ to score each component. For each component, the evaluator checks consistency between prediction and reference. If the component matches, the reward is set to 1; otherwise, if inconsistencies are detected, the reward is set to 0. The detailed prompt is shown in Figure 6.

C.2 Prompt for Fine-grained Evaluation

Figure 7 provides the prompt for the fine-grained fidelity evaluation in terms of content, structure, and style (line style, alignment, and cell style).

D Derivation of Aggregated Objective in CSPO

In this appendix, we provide a detailed derivation of Eq. (10) from Eq. (8) and (9).

D.1 Component-Level Objective Recap

Recall that CSPO decomposes the overall objective into component-specific terms:

$$\mathcal{L}(\theta) = \sum_{c \in \tilde{\mathcal{C}}} \mathcal{L}_c(\theta), \tag{13}$$

where each $\mathcal{L}_c(\theta)$ corresponds to the clipped policy gradient surrogate for component c:

$$\mathcal{L}_c(\theta) = \frac{1}{G}\sum_{g=1}^{G} \frac{1}{|\mathbf{y}_c^{(g)}|} \sum_{t=1}^{|\mathbf{y}_c^{(g)}|} \min\Big[\rho_{g,t}(\theta) A_{c,t}^{(g)},\ \text{clip}(\rho_{g,t}(\theta), 1-\epsilon, 1+\epsilon) A_{c,t}^{(g)}\Big], \tag{14}$$

$|\mathbf{y}_c^{(g)}|$ indicates the number of tokens associated with component c in rollout $\mathbf{y}^{(g)}$, ϵ denotes the clipping threshold.

D.2 From Component-Level to Unified Objective

To unify all component-level objectives into a single token-level surrogate, we first sum over all components:

$$\mathcal{L}(\theta) = \frac{1}{G}\sum_{g=1}^{G} \sum_{c \in \tilde{\mathcal{C}}} \frac{1}{|\mathbf{y}_c^{(g)}|} \sum_{t=1}^{|\mathbf{y}_c^{(g)}|} \min\Big[\rho_{g,t}(\theta) A_{c,t}^{(g)},\ \text{clip}(\rho_{g,t}(\theta), 1-\epsilon, 1+\epsilon) A_{c,t}^{(g)}\Big]. \tag{15}$$

Since each rollout $\mathbf{y}^{(g)}$ contains all tokens across components, we can reorganize the double sum over (c, t) into a single sum over all token indices

t in $\mathbf{y}^{(g)}$. To ensure proper weighting across components with different token spans, we normalize by the total token length $|\mathbf{y}^{(g)}|$:

$$\begin{aligned}
\mathcal{L}(\theta) &= \frac{1}{G}\sum_{g=1}^{G}\sum_{c\in\tilde{\mathcal{C}}}\frac{1}{|\mathbf{y}_c^{(g)}|}\frac{|\mathbf{y}^{(g)}|}{|\mathbf{y}^{(g)}|}\sum_{t=1}^{|\mathbf{y}_c^{(g)}|}\min\Big[\rho_{g,t}(\theta)A_{c,t}^{(g)}, \\
&\qquad \text{clip}(\rho_{g,t}(\theta),1-\epsilon,1+\epsilon)A_{c,t}^{(g)}\Big] \\
&= \frac{1}{G}\sum_{g=1}^{G}\frac{1}{|\mathbf{y}^{(g)}|}\sum_{c\in\tilde{\mathcal{C}}}\sum_{t=1}^{|\mathbf{y}_c^{(g)}|}\frac{|\mathbf{y}^{(g)}|}{|\mathbf{y}_c^{(g)}|}\min\Big[\rho_{g,t}(\theta)A_{c,t}^{(g)}, \\
&\qquad \text{clip}(\rho_{g,t}(\theta),1-\epsilon,1+\epsilon)A_{c,t}^{(g)}\Big] \\
&= \frac{1}{G}\sum_{g=1}^{G}\frac{1}{|\mathbf{y}^{(g)}|}\sum_{c\in\tilde{\mathcal{C}}}\sum_{t=1}^{|\mathbf{y}^{(g)}|}\frac{|\mathbf{y}^{(g)}|}{|\mathbf{y}_c^{(g)}|}\min\Big[\rho_{g,t}(\theta)A_{c,t}^{(g)}, \\
&\qquad \text{clip}(\rho_{g,t}(\theta),1-\epsilon,1+\epsilon)A_{c,t}^{(g)}\Big] \\
&= \frac{1}{G}\sum_{g=1}^{G}\frac{1}{|\mathbf{y}^{(g)}|}\sum_{t=1}^{|\mathbf{y}^{(g)}|}\min\Big[\rho_{g,t}(\theta)A_{t}^{(g)}, \\
&\qquad \text{clip}(\rho_{g,t}(\theta),1-\epsilon,1+\epsilon)A_{t}^{(g)}\Big],
\end{aligned} \tag{16}$$

where the aggregated token-wise advantage $A_t^{(g)}$ is as:

$$A_t^{(g)} = \sum_{c\in\tilde{\mathcal{C}}}\frac{|\mathbf{y}^{(g)}|}{|\mathbf{y}_c^{(g)}|}A_{c,t}^{(g)}. \tag{17}$$

Note that the second- to third- equation holds because $A_{c,t}^{(g)} = 0$ whenever $y_{g,t} \notin c$.

This formulation ensures that:

- **Balanced credit assignment:** Components with fewer associated tokens (e.g., global style indicators) are up-weighted, preventing their gradients from vanishing.
- **Unified training signal:** By aggregating all component-specific advantages into token-level surrogate, CSPO allows end-to-end optimization using a GRPO-style objective.

E Notation Summary

The main notations used in the CSPO framework are summarized in Table 5.

Table 5: Notation summary for CSPO framework.

Notation	Description
$\mathbf{x}$	Input table image
$\mathbf{y}$	Generated LaTeX sequence
$\mathbf{y}^*$	Reference LaTeX sequence
π_θ	Policy model with parameters θ
T	Length of LaTeX sequence
G	Number of rollouts per group
$\mathbf{y}^{(g)}$	The g-th rollout in a group
$\lvert\mathbf{y}^{(g)}\rvert$	Total token count in rollout g
$\mathcal{C}$	Set of functional components
$\tilde{\mathcal{C}}$	Extended component set including global
c	A specific component in $\mathcal{C}$ or $\tilde{\mathcal{C}}$
$\lvert\mathbf{y}_c^{(g)}\rvert$	Number of tokens in component c for rollout g
R_c	Component-specific reward for component c
$R_c^{(g)}$	Reward of component c in rollout g
$R_{\text{global}}^{(g)}$	Global reward (TEDS + compilation) for rollout g
$A_c^{(g)}$	Component-specific advantage for component c
$A_{c,t}^{(g)}$	Token-level component advantage at position t
$A_t^{(g)}$	Token-level aggregated advantage at position t
w_c	Weight for component c
α	Fixed weight for global component
β	Sensitivity parameter for adaptive weighting
λ	Smoothing parameter for weight updates
ϵ	Clipping threshold for PPO-style objective
$\mathbb{I}[\cdot]$	Indicator function
$\rho_{g,t}(\theta)$	Importance ratio $\pi_\theta(y_t^{(g)}\lvert y_{<t}^{(g)},\mathbf{x})/\pi_{\theta_{\text{old}}}(y_t^{(g)}\lvert y_{<t}^{(g)},\mathbf{x})$

F More Results

F.1 More Ablation

Effect of Specific Components. We study the influence of different component rewards. Table 6 shows that removing structure-related or style-related rewards leads to a significant performance drop, while the impact of removing the content-related reward is comparatively smaller. This is likely because content fidelity is already relatively high compared to structure and style fidelity (see the performance of GRPO). As a result, there is less room for further optimization on content, and incorporating content-specific rewards yields a smaller marginal effect. Removing all component-specific rewards reduces the model to the GRPO baseline, highlighting the importance of decomposed optimization.

Note that here the *content* reward (*i.e.*, caption and cell-appearance[2]) primarily affects text accuracy, the *structure* reward (*i.e.*, table layout) determines structure accuracy, the *style* reward (*i.e.*, column alignment and line style) influences style consistency.

Actually, the scheme CSPO w/o Style Reward only removes partial styles (*i.e.*, column alignment and line style) while still keeping the cell style. This is because in a cell, we cannot disentangle the textual content and style in considering a cell is already a small unit in a table. Then the cell content and style are still optimized. That may be

[2]In a cell, we do not disentangle the textual content and style in considering a cell is already a small unit in a table.

Variant	TEDS ↑	OF ↑	S ↑	C ↑	Y ↑	Y_{line} ↑	Y_{align} ↑	Y_{cell} ↑	R ↑
GRPO (no components)	87.7	42.0	74.7	86.7	50.3	74.9	71.5	92.0	99.6
CSPO w/o Content Reward	87.9	45.2	77.4	86.9	53.4	76.0	74.4	**92.5**	**99.7**
CSPO w/o Structure Reward	87.8	43.3	77.0	86.0	51.8	73.3	**74.5**	92.4	99.6
CSPO w/o Style Reward	87.7	43.1	**79.0**	**87.5**	49.6	72.3	72.7	92.4	99.6
CSPO (Ours)	**87.9**	**45.2**	77.6	87.0	**53.6**	**76.5**	74.2	92.1	99.6

Table 6: Ablation study on reward components on 3B models.

Variant	TEDS ↑	OF ↑	S ↑	C ↑	Y ↑	Y_{line} ↑	Y_{align} ↑	Y_{cell} ↑	R ↑
GRPO ($w_{global} = 10, w_{comp} = 0$)	87.7	42.0	74.7	86.7	50.3	74.9	71.5	92.0	99.6
CSPO ($w_{global} = 7, w_{comp} = 3$)	**88.0**	43.6	76.4	85.9	52.1	74.4	74.1	91.9	99.8
CSPO ($w_{global} = 5, w_{comp} = 5$)	87.9	44.8	77.1	86.9	53.1	76.0	73.8	**92.7**	99.7
CSPO ($w_{global} = 3, w_{comp} = 7$)	87.9	**45.2**	77.6	**87.0**	**53.6**	**76.5**	**74.2**	92.1	99.6
CSPO ($w_{global} = 0, w_{comp} = 10$)	87.7	44.7	**77.9**	86.3	51.2	73.7	73.4	92.1	**99.8**

Table 7: Ablation on reward weights for different components on 3B models.

Method	TEDS	OF	Structure	Content	Style
GRPO	87.7	42.0	74.7	86.7	50.3
CSPO (Graded 0-3)	**88.0**	44.4	77.3	86.7	51.8
CSPO (Binary 0/1)	87.9	**45.2**	**77.6**	**87.0**	**53.6**

Table 8: Performance comparison of CSPO using binary (0 or 1) and graded (0 to 3) reward on 3B models.

the reason that there is no performance drop on style fidelity.

Effect of Reward Weights. Table 7 shows the ablation study in different reward weight configurations, where w_{comp} denotes the sum of component-specific reward weights ($w_{comp} = \sum_{c \in \mathcal{C}} w_c$, with w_c equal for different $c \in \mathcal{C}$). CSPO consistently outperforms GRPO across all configurations, indicating that the effectiveness of CSPO does not rely on a specific weight setting. Among all configurations, $w_{comp} = 7$ achieves the best Overall Fidelity.

Effect of Reward Granularity. In our CSPO, by default we employ binary component-specific rewards. We further investigate the impact of reward granularity by introducing a graded scoring scheme. Specifically, instead of assigning a binary signal (0 or 1), each component is evaluated on a four-level scale (0–3), where 3 indicates perfect alignment, 2 denotes minor yet interpretable gaps, 1 corresponds to major errors, and 0 represents failed or invalid outputs. As shown in Table 8, compared with graded component-specific reward, the binary component-specific reward scheme achieves comparable TEDS and a higher Overall Fidelity. Both the two schemes obviously outperforms the GRPO baseline and we use binary reward by default.

F.2 Reliability of LLM-based Evaluation

In this work, we employ a strong LLM as an automatic judge to assess component-wise and overall fidelity. We use GPT-4o by default. Given concerns about evaluator bias and potential circularity, we conduct additional analyses on cross-model consistency, human alignment, and evaluation variance.

Cross-LLM Consistency. We evaluate model performance using multiple independent LLM-based judges in testing, including GPT-4o, Qwen3-Next-80B-A3B-Instruct, DeepSeek-v3.2, and GPT-5.2. Table 4 in the main manuscript and Table 9 here show that the overall performance trends are largely consistent across different LLM judges for 3B models and 7B models, respectively. In particular, CSPO consistently outperforms GRPO on most metrics. This indicates that our conclusions are not specific to a particular LLM judge, but instead reflect robust improvements in generation quality.

Human–LLM Agreement. We further assess the alignment between LLM-based judgments and human evaluation, where annotators label the overall fidelity (binary OF) of each output on a randomly sampled subset of 500 samples. Table 10 reports precision, recall, F1-score, and accuracy. GPT-4o and Qwen3-Next-80B-A3B-Instruct achieve strong agreement with human annotations, with F1-scores of 88.3 and 87.2, and accuracies of 89.8% and 88.8% (*i.e.*∼90%), respectively. DeepSeek-v3.2 and GPT-5.2 also show reasonable alignment, albeit with lower recall.

Overall, these results suggest that LLM-based

LLM Evaluators/Judge	Method	Overall Fidelity	Structure Fidelity	Content Fidelity	Style Fidelity
GPT-4o (by default)	SFT	50.7	79.7	90.3	59.0
	GRPO	50.6	80.7	89.7	58.9
	CSPO (Ours)	**53.0**	**81.4**	**90.0**	**60.6**
Qwen3-Next-80b	SFT	47.1	77.1	91.5	57.5
	GRPO	49.7	77.6	91.5	59.8
	CSPO (Ours)	**52.0**	**79.5**	**92.2**	**61.8**
DeepSeek-v3.2	SFT	43.3	79.3	90.4	51.4
	GRPO	45.8	80.5	90.3	54.7
	CSPO (Ours)	**47.8**	**81.6**	**91.0**	**56.2**
GPT-5.2	SFT	38.7	70.0	80.8	48.0
	GRPO	40.4	71.6	80.3	50.6
	CSPO (Ours)	**42.6**	**73.8**	**81.8**	**51.8**

Table 9: Performance comparisons among 7B SFT, GRPO and our CSPO models, with fine-grained metrics evaluated by using different LLM judges.

Evaluator	Precision	Recall	F1	Accuracy
GPT-4o	89.8	86.9	88.3	89.8
Qwen3-Next-80b	88.8	85.6	87.2	88.8
DeepSeek-v3.2	92.4	76.9	84.0	87.0
GPT-5.2	97.5	70.7	82.0	86.2

Table 10: Agreement between LLM judges with human annotations.

Method	OF	Structure	Content	Style
GRPO	42.0	74.7	86.7	50.3
CSPO (Reported)	45.2	77.6	87.0	53.6
CSPO (Mean ×8)	44.9	77.7	86.8	52.9
CSPO (Var ×8)	0.3	0.1	0.1	0.4

Table 11: Stability of GPT-4o evaluator in testing (8 independent runs) evaluated on 3B models.

evaluation closely aligns with human judgment, supporting its use as a reliable proxy in structured table-to-LaTeX generation.

Evaluation Variance and Stability. We assess the stability of LLM-based evaluation by repeating GPT-4o evaluation eight times on the same set of 4000 predictions from Qwen2.5-VL-3B-CSPO. As shown in Table 11, the low variance (0.1–0.4) across all metrics indicates that the LLM-based judge produces stable evaluation signals.

F.3 Training Dynamics

We compare the training dynamics of CSPO with GRPO. Figure 8 presents the reward curves for different components during training. For GRPO, we report component-specific rewards, although these rewards do not influence its optimization. Compared with GRPO, CSPO converges to higher rewards on Structure, Lines, Alignment, and Caption, while reaching a similar level for Cell Appearance.

F.4 More Visualization

Figure 9 to Figure 12 show more visualization that CSPO mitigates content, structure, and style errors. Figure 13 presents a failure case of CSPO on a complex table.

G More Discussion

G.1 Generalization to Other Tasks

We validate the proposed component-specific policy optimization framework on the table image-to-LaTeX generation task. The approach is conceptually generalizable to other structured generation problems (*e.g.*, HTML/CSS, code, and presentation generation).

Our framework includes a *domain-agnostic* credit assignment mechanism and a *domain-aware* component parser. The parser decomposes structured outputs into functional components, enabling localized reward attribution and optimization. Such decomposition is naturally supported in many domains through existing tooling—for example, DOM trees for HTML/CSS, abstract syntax trees (ASTs) for programming languages, and XML-based structures for document formats.

G.2 Training Cost

Training CSPO (2 epochs) requires 100+ million tokens in total. We view this as a one-time alignment cost. Actually, our framework is not tied to GPT-4o. It supports local open-weight LLM judges, eliminating API costs. For example, Qwen3-Next-80B achieves 90% agreement with human experts in evaluation and demonstrates evaluation trends highly consistent with GPT-4o, providing a scalable and cost-effective alternative for deployment.

Prompt for Rewarding

You are a reviewer specializing in LaTeX table code.
Please compare the following ground truth LaTeX table code and the predicted LaTeX table code, and assign a score to the predicted code.
Scoring Criteria and Corresponding Code Sections:
1. Alignment
Reference: \begin{tabular}{…}, including the type, order, and number of c, r, and l columns.
2. Vertical Lines
Reference: \begin{tabular}{…}, including '|' and '||'.
3. Horizontal Lines
Reference: Commands such as: \hline, \hline \hline, \cline, \hhline, \toprule, \midrule, \bottomrule, \cmidrule(lr){i-j}. The type, number, and position of the lines must match exactly.
4. Structure
Reference: Tokens such as &, \\, \multicolumn, and \multirow.
5. Caption
Reference: The content of \caption{…} and its position (whether it appears before or after the table).
6. Text Content and Style
Reference: The cell content and styling, excluding alignment, line types, structure, and caption. Minor formatting differences (e.g., 10 vs 10) should be considered correct. However, missing or incorrect content, or inconsistencies that affect semantics or formatting (e.g., \textbf{10} vs 10) should be considered incorrect.
7.Preamble
Reference: Only check commands before \begin{document} that are necessary for successful compilation, such as \usepackage{...}. Missing non-essential commands that do not affect compilability is acceptable and should not be penalized.

Each item should be scored as either 1 (correct) or 0 (incorrect), along with a reason justifying the score.
Please return your judgment as a JSON dictionary in the following format:

```
{
 "Alignment": {
  "reason": "...",
  "score": 1
 },
 "Vertical Lines": {
  "reason": "...",
  "score": 0
 },
 "Horizontal Lines": {
  "reason": "...",
  "score": 1
 },
 "Structure": {
  "reason": "...",
  "score": 1
 },
 "Caption": {
  "reason": "...",
  "score": 1
 },
 "Text Content and Style": {
  "reason": "...",
  "score": 1
 },
 "Preamble": {
  "reason": "...",
  "score": 0
 }
}
```

Ground truth code:
```latex
{gt_code}
```

Predicted code:
```latex
{pred_code}
```

Figure 6: Prompt for LLM-based rewarding for components of package dependencies, caption correctness, structural organization, cell appearance, column alignment, and rule placement (vline, hline).

Prompt for Evaluation

You will be given two LaTeX table codes: a Ground Truth version and a Predicted version.

Your task is to compare their rendered results across five aspects: Content, Structure, Line, Alignment and Cell Style. For each aspect, provide:
- "analysis": a short explanation of whether the prediction is correct
- "score": 1 if correct, 0 if there is any noticeable error

[EVALUATION CRITERIA]

1. Content
Check if all textual contents in the table are identical. Minor differences in LaTeX syntax (e.g., alternative math formatting) are acceptable as long as the rendered result is the same.

2. Structure
Check if the structure matches: row/column counts, merged cells (`\multicolumn`, `\multirow`), and cell positions.

3. Line
Check if horizontal and vertical lines are consistent in placement, numbers and style (e.g., `\hline`, `\hline \hline`, `\cline`, `\toprule`).

4. Alignment
Check if cell/column text alignment (left `l`, center `c`, right `r`) matches.

5. Cell Style
Check if text styles (e.g., bold, italic, underline, color) and background colors match.

[TABLE CODES]

Ground Truth Code:
```latex
{gt_code}
```

Predicted Code:
```latex
{pred_code}
```

[OUTPUT FORMAT]
Please return your judgment as a JSON dictionary like the following:

```
{
 "Content": {
  "analysis": "...",
  "score": 0
 },
 "Structure": {
  "analysis": "...",
  "score": 1
 },
 "Line": {
  "analysis": "...",
  "score": 0
 },
 "Alignment": {
  "analysis": "...",
  "score": 1
 },
 "Cell Style": {
  "analysis": "...",
  "score": 1
 }
}
```

Figure 7: Prompt for LLM-based fine-grained fidelity evaluation in terms of content, structure, and style (line style, alignment, and cell style).

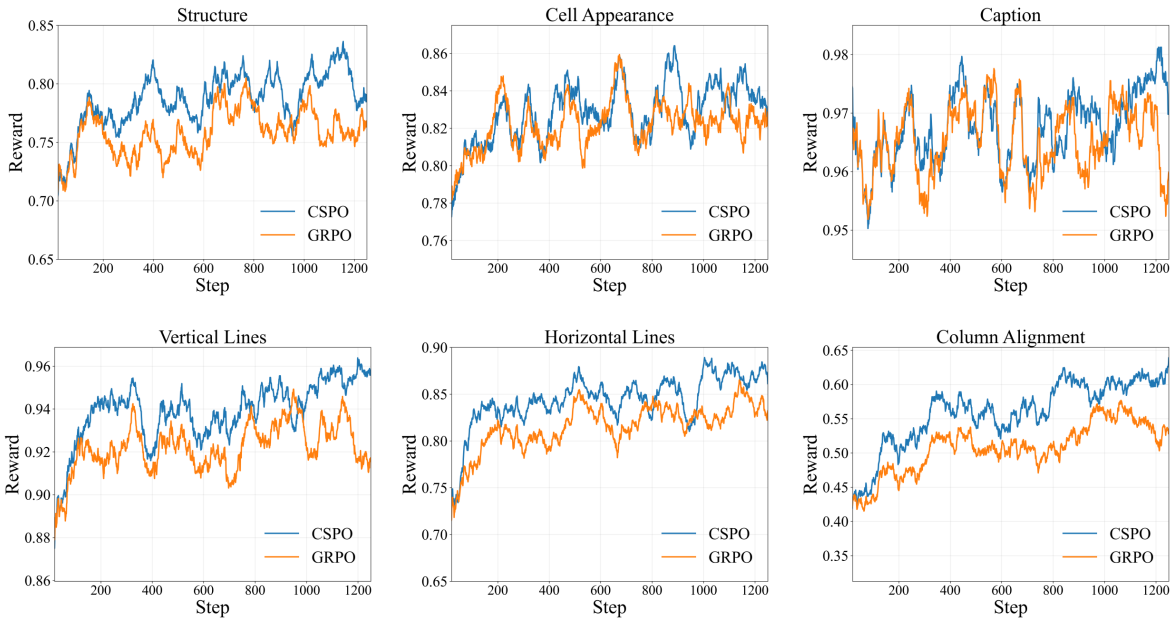

Figure 8: Training curves for 3B-GRPO and 3B-CSPO (ours). All curves are smoothed using a moving average (MA) for better visualization.

Ground Truth

Beam Width	Solution Length Nonbacktracking RW	Solution Length Simple RW
2**10	None	None
2**12	252	None
2**14	202	None
2**16	194	None
2**18	194	None
2**20	190 (Ideal)	None

Table 1: Without non-backtracking system cannot perform well on single epoch training. Effect of beam size is clear, but not so strong.

GRPO

Beam Width	Solution Length Nonbacktracking RW	Solution Length Simple RW
210	None	None
212	252	None
214	202	None
216	194	None
218	194	None
220	190 (Ideal)	None

Table 1: Without non-backtracking system cannot perform well on single epoch training. Effect of beam size is clear, but not so strong.

CSPO (Ours)

Beam Width	Solution Length Nonbacktracking RW	Solution Length Simple RW
2**10	None	None
2**12	252	None
2**14	202	None
2**16	194	None
2**18	194	None
2**20	190 (Ideal)	None

Table 1: Without non-backtracking system cannot perform well on single epoch training. Effect of beam size is clear, but not so strong.

Figure 9: A typical example comparing GRPO and CSPO of 3B models, showing CSPO mitigates **content errors**.

Ground Truth

	$k=-1$			
	t_μ^{-1}	$f_{1,\mu}^{-1}$	$f_{2,\mu}^{-1}$	$f_{3,\mu}^{-1}$
r	-3	-2	-2	-3
u	-2	-1	0	0
z	-1	0	0	-1
$\bar{z}$	-2	0	-1	0

Table 1: The three refined slices $\mathfrak{S}_{-1,f_i}$ that strictify $\mathfrak{Kin}^{(-1)}$.

GRPO

	$k=-1$			
	t_μ^{-1}	$f_{1,\mu}^{-1}$	$f_{2,\mu}^{-1}$	$f_{3,\mu}^{-1}$
r	-3	-2	-2	-3
u	-2	-1	0	0
z	-1	0	0	-1
$\bar{z}$	-2	0	-1	0

Table 1: The three refined slices $\mathcal{S}_{-1,f_i}$ that strictify $\mathfrak{kin}^{(-1)}$.

CSPO (Ours)

	$k=-1$			
	t_μ^{-1}	$f_{1,\mu}^{-1}$	$f_{2,\mu}^{-1}$	$f_{3,\mu}^{-1}$
r	-3	-2	-2	-3
u	-2	-1	0	0
z	-1	0	0	-1
$\bar{z}$	-2	0	-1	0

Table 1: The three refined slices $\mathfrak{S}_{-1,f_i}$ that strictify $\mathfrak{kin}^{(-1)}$.

Figure 10: A typical example comparing GRPO and CSPO of 3B models, showing CSPO mitigates **structure**, and **content** (in table caption) errors.

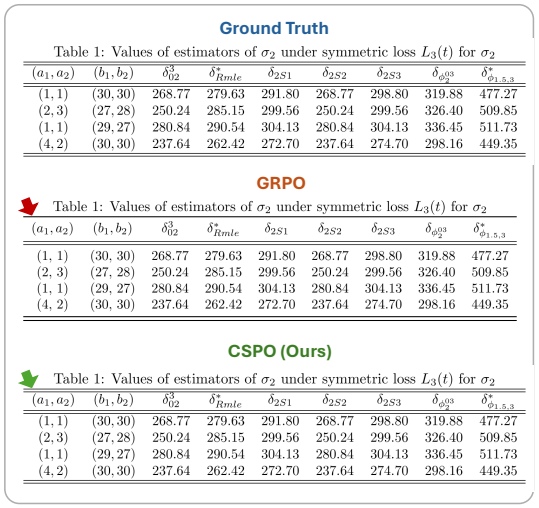

Figure 11: A typical example comparing GRPO and CSPO of 3B models, showing CSPO mitigates **line style errors**.

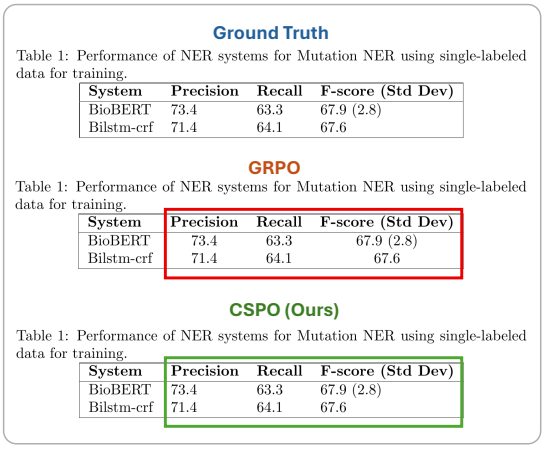

Figure 12: A typical example comparing GRPO and CSPO of 3B models, showing CSPO mitigates **alignment errors**.

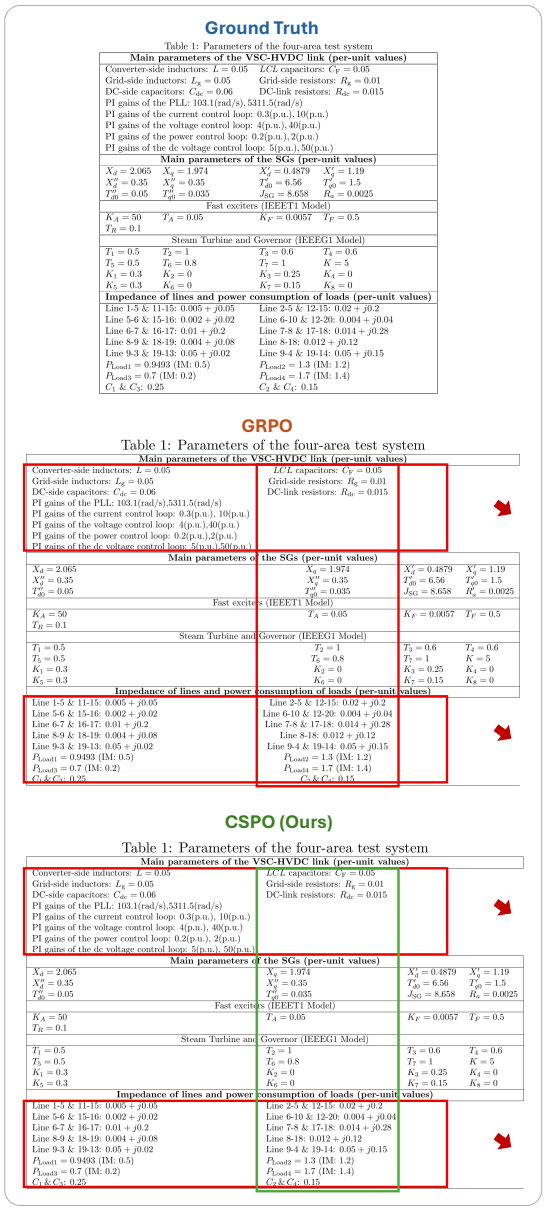

Figure 13: Failure case for GRPO and CSPO of 3B models. Both models' generations exhibit **structure errors** (marked by red boxes) on this complex table, where `\multicolumn{}` is used in groudtruth code but is ignored in the generated code. In addition, GRPO generation further has **alignment errors** (center aligned rather than left aligned as groudtruth).