

Aligning Text, Code, and Vision: A Multi-Objective Reinforcement Learning Framework for Text-to-Visualization

Mizanur Rahman^{‡*}, Mohammed Saidul Islam[‡], Md Tahmid Rahman Laskar[‡]
Shafiq Joty^{¶,§}, Enamul Hoque^{‡*}

[‡]York University, [¶]Salesforce AI Research, [§]Nanyang Technological University

Abstract

Text-to-Visualization (Text2Vis) systems translate natural language queries over tabular data into concise answers and executable visualizations. While closed-source LLMs generate functional code, the resulting charts often lack semantic alignment and clarity—qualities that can only be assessed post-execution. Open-source models struggle even more, frequently producing non-executable or visually poor outputs. Although supervised fine-tuning can improve code executability, it fails to enhance overall visualization quality, as traditional SFT loss cannot capture post-execution feedback. To address this gap, we propose RL-Text2Vis, the first reinforcement learning framework for Text2Vis generation. Built on Group Relative Policy Optimization (GRPO), our method uses a novel multi-objective reward that jointly optimizes textual accuracy, code validity, and visualization quality using post-execution feedback. By training Qwen2.5 models (7B and 14B), RL-Text2Vis achieves a 22% relative improvement in chart quality over GPT-4o on the Text2Vis benchmark and boosts code execution success from 78% to 97% relative to its zero-shot baseline. Our models significantly outperform strong zero-shot and supervised baselines and also demonstrate robust generalization to out-of-domain datasets like VIS-Eval and NVBench. These results establish GRPO as an effective strategy for structured, multimodal reasoning in visualization generation. We release our code at <https://github.com/vis-nlp/RL-Text2Vis>.

1 Introduction

Data visualization is central to understanding complex data, identifying patterns, and supporting data-driven decision-making (Rahman et al., 2025a; Aparicio and Costa, 2015; Hoque et al., 2022). However, creating accurate and interpretable visualizations from natural language queries requires

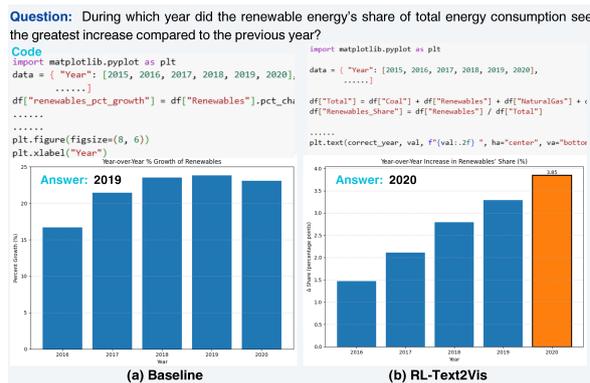


Figure 1: An example from Text2Vis (Rahman et al., 2025b). Given the data table and question as input, (a) the baseline model, Qwen2.5-14B-Instruct generates runnable visualization code, but the visualization is not aligned with the query as it shows the growth of renewables quantity instead of the share of total, leading to an incorrect answer. (b) Our RL-Text2Vis-14B produces a correct, query-aligned, and interpretable visualization.

programming expertise, familiarity with visualization libraries, and design knowledge—posing a barrier for non-technical users (Shen et al., 2022). Alongside visualizations, concise textual answers are often crucial for quickly validating results and providing context, as users often seek both a summary insight and a visual representation (Rahman et al., 2025b). For example, in Figure 1, the textual answer “2020” identifies the correct year with the greatest increase in renewables share, helping users interpret the chart without manually extracting values. Text-to-Visualization (Text2Vis) addresses this challenge by automatically converting natural language queries over tabular data into concise answers and accurate visualizations, thereby eliminating the need for manual coding.

Despite advances in large language models (LLMs) and vision-language models (VLMs), their ability to generate high-quality visualizations remains limited. Benchmarks such as VisEval (Chen et al., 2024) and Text2Vis (Rahman et al., 2025b) reveal persistent failures in semantic correctness,

* Contact Emails: {mizanurr, enamulh}@yorku.ca

code executability, and chart readability. Closed-source models like GPT-4o, though strong in code generation, often produce visually misaligned charts and raise compliance concerns (Laskar et al., 2023, 2024). Open-source models perform even worse; small to mid-size versions (7B-14B) favored for on-premise deployments frequently produce non-executable or misaligned outputs, while larger models achieve only moderate performance and remain too expensive for many real-world applications (Mallick et al., 2024; Chen et al., 2024).

Unlike code generation, where aspects of functional correctness can often be verified by execution success (i.e., the program runs without exceptions), Text2Vis introduces additional quality dimensions that are only observable after rendering, notably chart readability, semantic alignment with the query, and visual clarity. Current approaches typically use supervised fine-tuning (SFT) that minimizes token-level loss to imitate reference outputs. While this improves executability by aligning syntax and structure, it cannot optimize these post-execution qualities because traditional loss functions provide no signal about visualization clarity or semantic alignment. This limitation highlights the need for alternative training strategies that incorporate post-execution feedback (Wei et al., 2025).

Meanwhile, Reinforcement learning (RL) has emerged as a powerful technique for improving reasoning in LLMs, with models like DeepSeek-R1 (Guo et al., 2025) and methods like SWE-RL (Wei et al., 2025) demonstrating significant gains on coding and software engineering tasks. However, these methods primarily rely on single-modal feedback such as numeric correctness or code executability. Data visualization presents a more complex challenge, as its quality depends on a combination of textual accuracy, query alignment, code validity, and the visual clarity of the rendered chart. Effectively optimizing for these multimodal requirements necessitates a post-execution, multi-objective reward that can jointly align the model’s reasoning, code, and final visual output.

In this work, we propose RL-Text2Vis, the first RL framework tailored for the Text2Vis tasks. It uses Group Relative Policy Optimization (GRPO) (Wang et al., 2023) to incorporate post-execution feedback, enabling optimization beyond code correctness to include visualization clarity and semantic alignment. Unlike existing RL-based methods that rely on single-modal signals, RL-Text2Vis introduces a novel multi-objective reward that jointly

considers three aspects critical for visualization: (i) syntactic and functional validity of the code, (ii) alignment, correctness, and quality of the generated chart, and (iii) textual correctness of the predicted answer. This design explicitly captures the multimodal nature of the problem, addresses the shortcomings of SFT and single-modal RL methods, and improves generalization across diverse data contexts and query types.

We implement RL-Text2Vis on top of Qwen2.5 Instruct models of size 7B and 14B (Yang et al., 2024), and train them using GRPO with our proposed multi-objective reward. We conduct extensive experiments on the Text2Vis benchmark (Rahman et al., 2025b) and further evaluate generalization on VIS-Eval (Chen et al., 2024) and NVBench (Luo et al., 2021). Our RL-Text2Vis outperforms SFT baselines, zero-shot open-source models, and even proprietary systems like GPT-4o, achieving over 22% relative improvement compared to GPT-4o in both chart clarity and correctness. These results demonstrate that RL with post-execution visual feedback is an effective strategy for structured, multimodal reasoning in visualization generation. Moreover, RL-Text2Vis offers a practical, deployable approach for real-world scenarios where privacy, cost, and compliance constraints limit the use of closed-source models.

In summary, our contributions are: (i) We propose RL-Text2Vis, the first RL framework for Text-to-Visualization, optimizing query-aligned and interpretable visualizations. (ii) We introduce a novel multi-objective reward leveraging post-execution feedback to jointly optimize visualization clarity and alignment, code validity, and textual correctness. (iii) Our approach significantly outperforms all baselines, achieving 22% higher chart clarity and correctness than GPT-4o and improving code executability to 97% on the Text2Vis benchmark. (iv) RL-Text2Vis demonstrates strong out-of-domain generalization on VIS-Eval and NVBench, proving robustness across diverse queries and domains.

2 Related Work

LLMs and VLMs for Visualization Early systems for text-to-visualization generation, such as NL4DV (Narechania et al., 2020) and Advisor (Liu et al., 2021), relied on rule-based grammars or template-driven specifications. While these approaches ensured syntactic correctness, they were

rigid, lacked scalability, and could not handle the diversity of real-world analytical queries. Recent advances in LLMs have enabled more flexible approaches for text-to-visualization generation. Methods such as Chat2VIS (Maddigan and Susnjak, 2023), LIDA (Dibia, 2023), ChartGPT (Tian et al., 2024), and ChatVis (Mallick et al., 2024) translate natural language queries into executable visualization code through prompting, while Prompt4Vis (Li et al., 2024) enhances schema-aware prompting for improved accuracy. ChartLlama (Han et al., 2023) leverages instruction tuning to improve chart reasoning and generation quality. Frameworks like MDSF (Zhang et al., 2025) extend this paradigm to multi-dimensional data storytelling by integrating contextual insights with automated visualization. Others have adopted agentic frameworks to jointly generate visualizations and text as coherent data stories (Islam et al., 2024).

In parallel, vision-language models (VLMs), particularly those adapted to chart-centric tasks through fine-tuning or fine-tuning followed by reinforcement learning (Masry et al., 2023, 2024a,b, 2025), have significantly advanced *static chart understanding and reasoning*, such as question answering (Masry et al., 2022; Kantharaj et al., 2022a) and chart summarization given an existing chart (Kantharaj et al., 2022b). These approaches focus on interpreting existing visualizations and do not address *visualization generation*.

Despite these advances, LLM-based methods still face critical challenges. They often produce incorrect or incomplete answers, fail to generate executable code, or create charts that lack clarity or relevance to the analytical intent (Chen et al., 2024; Rahman et al., 2025b). These limitations indicate that supervised fine-tuning and prompt engineering alone are insufficient: SFT optimizes only token-level likelihoods, often improving executability via syntax/library alignment. However, it cannot target post-execution qualities such as visual clarity. Prompting, by design, does not update model parameters (Zhang et al., 2024). Consequently, neither incorporates the multimodal feedback available after execution that is needed for interpretable, trustworthy visualizations. We address this by optimizing a post-execution, multi-objective reward.

Visualization Evaluation Evaluating visualization generation has traditionally relied on rule-based metrics, which fail to capture semantic alignment or visual quality, or on manual evaluation, which is labor-intensive and cannot scale (Liu et al., 2021;

Srinivasan et al., 2021). Recent work shows that LLMs and vision-language models (VLMs) can serve as reliable evaluators by reasoning jointly over text, code, and images (Chen et al., 2024; Rahman et al., 2025b). Closed-source models such as GPT-4o and Gemini (Gu et al., 2024) provide strong evaluation capabilities, while open-source alternatives like Qwen-VL and LLaVA offer practical solutions for privacy-constrained environments (Laskar et al., 2025). These models assess both textual correctness and visual quality, making them suitable as reward models. However, existing evaluation methods are primarily used for post-hoc benchmarking rather than model optimization. We integrate multimodal, post-execution feedback (text, code, image) directly into the training loop by converting LLM/VLM evaluators into a multi-objective RL reward.

RL for Code Generation and Reasoning RL has emerged as a powerful approach for improving reasoning and structured outputs in LLMs (Tie et al., 2025). Techniques such as RLHF (Ouyang et al., 2022) and Direct Preference Optimization (DPO) (Rafailov et al., 2023) align models with human preferences with respect to safety, factual correctness and stylistic aspects such as formats rather than structured reasoning or execution correctness. Proximal Policy Optimization (PPO) based approaches (Schulman et al., 2017) use rule-based rewards (e.g., execution success, exact answer matching) but require a critic model, making them computationally expensive.

Recent work shows RL from verifiable rewards improves LLM reasoning. DeepSeek-R1 (Guo et al., 2025) introduced GRPO, a scalable policy-gradient method that removes the learned critic by ranking multiple outputs per prompt and computing relative advantages. SWE-RL (Wei et al., 2025) applied GRPO to software engineering tasks, showing strong results in structured, code-intensive domains, yet these methods rely on single-modal feedback such as code execution success or numeric correctness. CodeRL (Le et al., 2022) optimizes code synthesis via functional correctness and unit-test rewards, while ChartGPT (Tian et al., 2024) addresses chart generation through supervised or heuristic feedback.

Text-to-Visualization requires optimizing multiple post-execution objectives: semantic correctness and query alignment, code validity and executability, and chart readability. To our knowledge, no prior work integrates post-execution mul-

timodal feedback into RL or combines it with a multi-objective reward balancing textual, code, and visualization quality. Our RL-Text2Vis framework uses GRPO with this dual novelty, enabling scalable optimization across all visualization dimensions without training a value network.

3 RL-TEXT2VIS

We introduce **RL-Text2Vis**, the first RL framework for Text2Vis generation that leverages post-execution, multimodal feedback, since visualization quality is only observable after rendering. The pipeline, illustrated in Figure 2, operates as follows: given a natural language query and a table, the policy produces a structured output containing a concise answer and visualization code. Each output is scored by a two-stage reward: (1) a format reward enforcing structural validity and (2) a composite multimodal reward. Policy updates are performed with GRPO, which provides efficient training without a learned critic model, while directly aligning the model to high-level quality objectives.

3.1 Problem Formulation

We formalize Text2Vis generation as mapping an input $x = (\text{query}, \text{table}) \in \mathcal{D}$ to an output $y = (\text{answer}, \text{code}) \sim \pi_\theta(\cdot | x)$, where π_θ denotes the policy with parameters θ . After executing the generated code to render a chart, a scalar reward $R(x, y)$ is assigned using a two-stage mechanism that combines structural and quality-based signals (see Section 3.3). Under this RL formulation, the objective is to maximize the expected reward:

$$\max_{\theta} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta(\cdot | x)} [R(x, y)], \quad (1)$$

Unlike SFT, which minimizes the negative log-likelihood of reference outputs (i.e., off-policy), this objective is on-policy and directly targets multi-objective, post-execution criteria such as visualization clarity, semantic alignment, and code executability that cannot be expressed through token-level losses.

3.2 GRPO Optimization

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left(\min \left(i_t(\theta) \hat{A}_{i,t}, \text{clip} \left(i_t(\theta), 1 - \varepsilon, 1 + \varepsilon \right) \hat{A}_{i,t} \right) - \beta D_{\text{KL}}[\pi_\theta \| \pi_{\text{ref}}] \right) \right], \quad (2)$$

GRPO (Shao et al., 2024) is a scalable policy gradient method that eliminates the need for a learned value function with group-based relative

advantages obtained by “ranking” multiple samples per prompt. This critic-free, low-variance update scales well to long sequences where value estimation could be impractical. Unlike PPO (Schulman et al., 2017), which relies on a critic, GRPO uses reward-standardized advantages together with PPO-style clipping and a KL penalty to a frozen reference policy, yielding efficient optimization.

For each prompt q , the policy π_θ generates a group of G candidate outputs $\{y_1, y_2, \dots, y_G\}$. A composite reward is computed for each output using our multi-objective reward function (Section 3.3), and rewards are standardized within the group to compute the advantage \hat{A}_i . Specifically:

$$\hat{A}_i = \frac{r_i - \bar{r}}{\sigma_r}, \quad (3)$$

where r_i is the reward for y_i , \bar{r} and σ_r are the group mean and standard deviation of rewards. Intuitively, outputs above the group average receive positive advantages, while those below receive negative ones.

The complete GRPO objective, which combines ratio clipping for stable policy updates and a KL regularization term to prevent excessive divergence from the reference policy, is shown in Equation 2, where $i_t(\theta) = \frac{\pi_\theta(y_{i,t} | q, y_{i,<t})}{\pi_{\text{old}}(y_{i,t} | q, y_{i,<t})}$ is the importance sampling ratio between the current (trainable) policy and the old frozen policy (previous iterate) for the t -th token in y_i . $\hat{A}_{i,t}$ is the within-group, normalized advantage as per Eq. 3 (every token in y_i gets the same advantage); π_{ref} is the fixed reference policy used in the KL term. Finally, ε sets the PPO-style clipping bound on the importance ratio $i_t(\theta)$ to stabilize updates, and β weights the KL penalty $D_{\text{KL}}[\pi_\theta | \pi_{\text{ref}}]$ to limit divergence from the reference policy.

3.3 Multi-Objective Reward Design

A key challenge in Text2Vis generation lies in jointly optimizing correctness across text, code, and visualization dimensions. To address this, we adopt a two-stage reward system that ensures structural validity while providing fine-grained feedback for quality improvement.

Stage 1: Format Reward. The format reward enforces compliance with the required output schema, where the model must return a valid JSON object containing two fields: `answer` (a short textual response) and `code` (a runnable Python visualization script ending with `plt.show()`). Responses failing this structural constraint receive a reward of

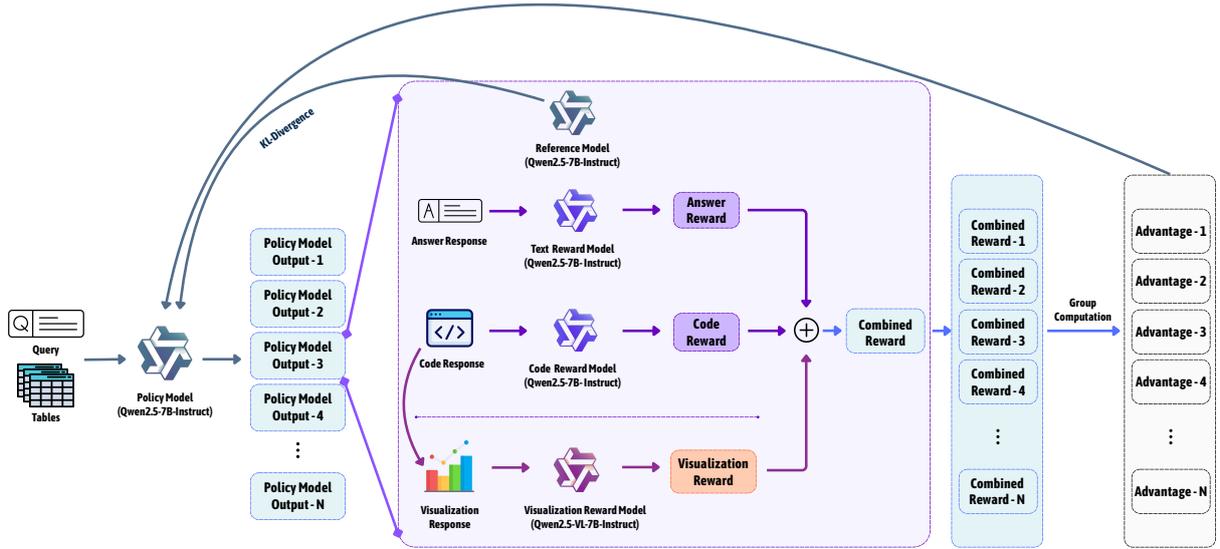


Figure 2: GRPO training architecture for text-to-visualization generation, showing policy outputs, multi-objective rewards (answer, code, visualization), combined reward computation, and advantage calculation for policy updates.

zero and are excluded from subsequent optimization. This step prevents policy drift toward invalid generations.

Stage 2: Composite Reward. For responses passing the format check, we compute a composite reward integrating three complementary signals:

1. **Textual Correctness (R_{text}):** We use an LLM-based evaluator to assess semantic alignment between the generated answer and the ground truth. The score ranges from 0 to 1, with partial credit for near-synonyms or numerically close values.
2. **Code Reward (R_{code}):** This combines two aspects with equal weight: (i) *Executability*, verified by running the generated code in a sandboxed environment, and (ii) *Intent Match*, scored by an LLM comparing generated code with reference query and code. We compute $R_{\text{code}} = \frac{1}{2}I_{\text{exec}} + \frac{1}{2}I_{\text{intent}}$ with $I_{\text{exec}}, I_{\text{intent}} \in \{0, 1\}$, so $R_{\text{code}} \in \{0, 0.5, 1\}$.
3. **Visualization Quality (R_{vis}):** We leverage a VLM to evaluate the resulting chart on two dimensions: readability (layout clarity, label quality) and correctness (faithfulness to query and data). Both scores are normalized to $[0, 1]$ and averaged.

The final composite reward is expressed as a weighted sum:

$$R = \alpha R_{\text{text}} + \beta R_{\text{code}} + \gamma R_{\text{vis}}, \quad \text{with } \alpha + \beta + \gamma = 1, \quad (4)$$

where α, β, γ control the relative emphasis on textual, code, and visual quality. The weight-

ing strategy is a practical tuning step to stabilize training. We performed a small grid search over $\alpha \in \{0.3, 0.4, 0.5, 0.6\}$, $\beta \in \{0.2, 0.25, 0.3\}$, and $\gamma = 1 - \alpha - \beta \geq 0.1$, and selected $(\alpha, \beta, \gamma) = (0.50, 0.25, 0.25)$ as the *best trade-off* setting: it improved code executability and visualization quality without degrading textual correctness. This choice maximized the overall validation metrics, indicating balanced performance across textual, code, and visual dimensions.

3.4 Implementation Details

We trained two open-source models, Qwen2.5-7B-Instruct and Qwen2.5-14B-Instruct, using the proposed RL-Text2Vis framework on the Text2Vis benchmark (Rahman et al., 2025b). The Text2Vis dataset comprises two splits: *test1* with 1,749 samples used for RL optimization, and *test2* with 236 stratified samples used for final evaluation.

GRPO was run with a group size of $G = 8$ (completions per prompt) with within-group, reward-based advantage normalization. With a per-device batch of 8 prompts on 6 GPUs and gradient accumulation of 8, the rollout batch is 48 groups per micro-step and the effective batch is 384 groups per optimizer update. Rewards were computed using the two-stage scheme in Section 3.3. Specifically, the text-reward judge was Qwen2.5-7B for the 7B policy and Qwen2.5-14B for the 14B policy; the visual-reward judge was Qwen2.5-VL-7B and Qwen2.5-VL-32B, respectively. These judge models are used in an off-the-shelf manner, prompted without any fine-tuning. To mitigate evaluator

bias, we conducted a stratified 200-sample comparison of rewards from Qwen2.5-7B, Qwen2.5-VL-7B, and GPT-4o, and observed strong cross-judge agreement (Pearson $r = 0.85\text{--}0.97$) (see Appendix A.2 for full details).

For our selection of base models, we excluded coding models whose performance on Text2Vis were substantially worse, as this benchmark requires not only code generation but also reasoning over flattened tabular data and producing semantically aligned textual answers. Among open-source instruct models, Qwen2.5 achieved the strongest zero-shot performance on Text2Vis, motivating its selection as the primary backbone. To verify framework generality, we also trained a Llama-3.1-8B (Grattafiori et al., 2024) model under the same GRPO setup, which yielded comparable results and confirmed the framework’s model-agnostic applicability 5.5.

4 Experiments

4.1 Benchmarks

We evaluate RL-Text2Vis on four benchmarks covering in-domain and out-of-domain settings:

(i) Text2Vis (In-Domain): This benchmark is designed for natural language to visualization generation and includes natural language queries, tabular datasets, and reference outputs (answer and visualization code). Following the official evaluation split, *test2* (236 stratified samples) is used strictly as a held-out evaluation set; no hyperparameters are tuned on *test2*.

(ii) VisEval (Out-of-Domain): A benchmark with 2,524 queries across 146 databases, designed to evaluate visualization systems on validity, legality, and readability (Chen et al., 2024). It features ambiguous mappings, challenging layouts, and diverse schemas. It does not provide ground-truth textual answers, so we report only code executability and visual quality (readability, correctness).

(iii) NVBench (Out-of-Domain): A large-scale dataset with 7,247 visualization generation tasks across 105 domains and 153 datasets (Luo et al., 2021). It supports evaluation on complex analytical queries and diverse chart types. Like VisEval, it does not include ground-truth textual answers, so Answer Correctness is omitted.

(iv) PandasPlotBench (Out-of-Domain): A benchmark of 175 data points designed to assess a model’s ability to generate executable and visually accurate plotting code from Pandas DataFrame

(Galimzyanov et al., 2025).

4.2 Baselines

We compare RL-Text2Vis against a comprehensive set of baselines covering closed-source and open-source LLMs, code-specialized models, and supervised fine-tuned variants. Among proprietary systems, GPT-4o (OpenAI, 2024) and Gemini 1.5 Flash and 2.0 Flash (Team, 2024) serve as state-of-the-art commercial models and strong zero-shot baselines. For open-source alternatives, we include LLaMA-3.1-8B, CodeLLaMA (7B and 13B) (Roziere et al., 2023), and Mistral-7B (Jiang et al., 2023). We also evaluate Qwen2.5 models (Yang et al., 2024) in both general-purpose and code-optimized (Coder-7B) variants under zero-shot settings to assess raw capability. In addition, we include supervised fine-tuned baselines trained on the Text2Vis *test1* split (5 epochs): Qwen2.5-7B SFT and Qwen2.5-14B SFT. We exclude DPO because it requires preference data, which is not readily available for Text2Vis. In our preliminary experiments, we also found GRPO to be more efficient—since it does not require a value model—and more effective than PPO. Therefore, we adopt GRPO as our RL algorithm. All systems (zero-shot baselines, the SFT model, and our RL-Text2Vis models) are decoded with temperature = 0.7, top- p = 0.9, and a 2,048-token limit to ensure comparability.

4.3 Evaluation Metrics

We follow the evaluation protocol introduced in (Rahman et al., 2025b) to assess text-to-visualization generation across four dimensions: (i) **Answer Correctness**, which measures whether the generated textual response matches the ground truth; (ii) **Code Executability**, which checks if the visualization code runs without errors; (iii) **Chart Readability**, which evaluates layout clarity and labeling quality; and (iv) **Chart Correctness**, which verifies whether the visualization accurately reflects the intended analytical insight. Answer correctness and code executability are treated as binary metrics, while chart readability and correctness are rated on a 1–5 scale. A sample is considered a pass if the code executes successfully, the answer matches the ground truth, and both the readability and chart correctness scores are at least 3.5. We use GPT-4o as the primary evaluator, as it has shown strong agreement with human ratings for both textual and visual assessments (Chen et al., 2024; Gu et al., 2024). To reduce evaluation bias,

we perform manual evaluation on the official evaluation set (236 samples) across RL-Text2Vis (7B, 14B), Qwen2.5 (7B, 14B) zero-shot counterparts, GPT-4o, and the Qwen2.5 (7B, 14B) SFT baselines. See evaluation guidelines and metric definitions in Appendix Table 6.

5 Results

5.1 In-Domain Results

Table 1 presents in-domain results on the Text2Vis benchmark. RL-Text2Vis substantially outperforms all open-source baselines across every metric. For example, the RL-Text2Vis 14B model improves chart readability from 3.12 to 4.10 and chart correctness from 2.94 to 4.03 compared to zero-shot Qwen2.5-14B. Similarly, code execution success rises from 78% to 97%, and answer match improves from 29% to 35%, demonstrating the benefit of multimodal reward optimization. In contrast, supervised fine-tuning offers only marginal gains over zero-shot models, highlighting the limitations of standard instruction tuning in visualization tasks. The RL-Text2Vis 7B model also shows large improvements, achieving a chart readability score of 3.84 compared to 2.81 and chart correctness of 3.86 compared to 2.69 for its zero-shot counterpart.

Notably, RL-Text2Vis closes the gap with proprietary models and even surpasses them on key visualization metrics. While the state-of-the-art GPT-4o model achieves a comparable final pass rate, RL-Text2Vis (7B and 14B) outperforms it in chart readability, visual clarity, and correctness. For example, the RL-Text2Vis 14B model achieves a chart readability score of 4.10 compared to 3.32 and chart correctness of 4.03 compared to 3.30, showing a clear margin. These results demonstrate that RL with multimodal feedback enables open-source models to match or even exceed closed-source systems in visualization quality.

5.2 Out-of-Domain Generalization

Table 2 reports out-of-domain performance on VIS-Eval and NVBench, where models are trained only on the Text2Vis *test1* set and evaluated without further adaptation. RL-Text2Vis demonstrates strong generalization across both benchmarks, substantially outperforming zero-shot Qwen baselines. On VIS-Eval, which is particularly challenging due to multi-table schemas, RL-Text2Vis-7B improves chart readability from 1.50 to 2.50 and chart correctness from 0.69 to 1.37, along with a notable

gain in code execution success (from 57% to 72%).

On NVBench, which includes large-scale cross-domain tasks, RL-Text2Vis achieves even greater improvements. Code execution success improves from 75% to 93%, while chart readability rises from 2.64 to 3.47 and chart correctness from 2.34 to 3.28. On PandasPlotBench, which evaluates executable plotting from Pandas DataFrames, RL-Text2Vis maintains strong generalization, further validating its robustness across diverse visualization domains. These results show that optimizing semantic correctness, code validity, and visual quality during RL training improves generalization beyond the training domain, including to datasets with different schemas and chart types.

5.3 Human Evaluation

While GPT-4o serves as our primary evaluator due to its strong alignment with human judgments (Gu et al., 2024; Rahman et al., 2025b), we also conducted a manual study to ensure robustness. Two annotators independently assessed the full stratified official evaluation set of the Text2Vis benchmark (236 samples) using a structured rubric (Table 6) covering label clarity, color use, font size, and overall visual appeal. They evaluated outputs from all major baselines—RL-Text2Vis (7B/14B), Qwen2.5 (zero-shot/SFT), and GPT-4o—while blinded to automated scores. Agreement between human and automated judgments was consistently strong, with Pearson correlation coefficients of $r = 0.91$ for Answer Match, $r = 0.88$ for Clarity & Readability, and $r = 0.88$ for Chart Correctness. In addition, code executability is measured objectively by verifying whether the generated Matplotlib code executes without errors, making this dimension independent of both human and automated judgments. These results confirm that our reported improvements are reliably supported by both human and automated evaluation.

5.4 Ablation Studies

Effect of Individual Reward Signals. We perform ablations on the RL-Text2Vis-7B model to analyze the contribution of each reward component and their combination. Table 3 reports two complementary analyses: removal-based ablations, where each component is dropped from the full model, and single-component ablations, where each reward is used independently. The results clearly show that using the multimodal reward yields the highest overall performance, confirming the ef-

Model	Code Exec. Success (%)	Answer Match (%)	Visual Clarity Readability	Chart Correctness	Final Pass Rate (%)
GPT-4o (Zero-Shot)	87	39	3.32	3.30	30
Gemini-1.5-Flash (Zero-Shot)	82	32	3.26	2.95	16
Gemini-2.0-Flash (Zero-Shot)	90	35	3.73	3.66	26
Claude-Sonnet-4	82	31	3.67	3.54	19
Mistral-7B (Zero-Shot)	42	22	1.57	1.39	7
Llama-3.1-8B (Zero-Shot)	70	25	1.81	1.62	8
CodeLlama-7B (Zero-Shot)	62	11	2.34	1.50	2
CodeLlama-13B (Zero-Shot)	54	16	1.81	1.45	4
Qwen-2.5-Coder-7B (Zero-Shot)	32	24	1.37	1.23	4
Qwen-2.5-7B (Zero-Shot)	77	27	2.81	2.69	13
Qwen-2.5-14B (Zero-Shot)	78	29	3.12	2.94	14
Qwen-2.5-7B (SFT)	85	30	3.34	3.32	16
Qwen-2.5-14B (SFT)	87	36	3.42	3.28	18
RL-Text2Vis-7B (Gen-8)	91	31	3.84	3.86	22
RL-Text2Vis-14B (Gen-8)	97	35	4.10	4.03	29

Table 1: Performance on Text2Vis. Readability and correctness are rated on a 1–5 scale. Final Pass Rate (%) reflects combined criteria.

Model	Code Exec. Success (%)	Visual Clarity Readability	Chart Correctness
VIS-Eval			
Qwen-2.5-7B (Zero-Shot)	57	1.50	0.69
Qwen-2.5-7B (SFT)	64	1.86	0.87
RL-Text2Vis-7B (Gen-8)	72	2.50	1.37
RL-Text2Vis-14B (Gen-8)	74	2.58	1.48
NVBench			
Qwen-2.5-7B (Zero-Shot)	75	2.64	2.34
Qwen-2.5-7B (SFT)	82	3.07	2.79
RL-Text2Vis-7B (Gen-8)	93	3.47	3.28
RL-Text2Vis-14B (Gen-8)	96	3.95	3.59
PandasPlotBench			
Qwen-2.5-7B (Zero-Shot)	65	2.42	2.49
RL-Text2Vis-7B (Gen-8)	75	3.32	3.37
RL-Text2Vis-14B (Gen-8)	79	3.65	3.63

Table 2: Out-of-domain results on VIS-Eval, NVBench & PandasPlotBench (readability, correctness: 1–5)

Configuration	Code Exec. Success (%)	Answer Match (%)	Visual Clarity Readability	Chart Correctness	Final Pass Rate (%)
Full (All Rewards)	91	31	3.84	3.86	22
- Format Reward	87	25	3.38	3.25	17
- Answer Reward	89	24	3.79	3.82	19
- Code & Visual Reward	82	28	2.98	3.10	13
Format Reward Only	78	25	3.02	2.88	13
Answer Reward Only	76	30	2.72	2.63	11
Code Reward Only	86	26	2.92	2.81	14
Visual Reward Only	79	26	3.22	3.05	14

Table 3: Ablation on reward components (Qwen2.5-7B) on Text2Vis.

fectiveness of jointly optimizing textual accuracy, code validity, and visualization quality.

Effect of Number of Sampled Completions. To assess the effect of sample size during GRPO optimization, we trained RL-Text2Vis-7B model with 4 versus 8 completions per input. Table 5 shows that using more completions stabilizes ranking-based updates and yields consistent gains. These findings suggest that sampling more candidate outputs per input during preference optimization provides stronger learning signals, yielding more reliable and interpretable charts.

5.5 Scaling and Cross-Architecture Analysis

To assess scalability and model generalizability, we trained RL-Text2Vis on both smaller (3B) and alter-

Configuration	Code Exec. Success (%)	Answer Match (%)	Visual Clarity Readability	Chart Correctness	Final Pass Rate (%)
Qwen2.5-3B (Zero-Shot)	67	13	1.94	1.63	3
RL-Text2Vis-3B	88	12	2.23	1.91	4
Llama-3.1-8B (Zero-Shot)	70	25	1.81	1.62	8
RL-Text2Vis-Llama-3.1-8B	87	28	2.91	2.67	15

Table 4: Scaling and Cross-Architecture Results on Text2Vis.

native (Llama-3.1-8B) architectures under the same GRPO setup. As shown in Table 4, applying RL consistently improved all metrics across scales and model families. For Qwen2.5-3B, RL-Text2Vis increased code executability from 67% to 88% and enhanced chart readability and correctness; however, textual answer accuracy remained unchanged, revealing that smaller models can learn syntactic and visual patterns but lack sufficient capacity for reasoning improvements. Similarly, on Llama-3.1-8B, RL-Text2Vis improved all metrics, demonstrating effective transfer beyond the Qwen architecture. These results confirm that our GRPO-based framework is architecture-agnostic and scales robustly across model families and sizes.

6 Error Analysis

We manually evaluated all samples generated by the zero-shot Qwen2.5-14B baseline and our RL-Text2Vis-14B model to analyze failure patterns. Common failures included *syntax errors* (e.g., “invalid syntax” or missing commas), *shape mismatches* (e.g., “x and y must have same first dimension”), and *value errors* such as incorrect CAGR or percentage calculations. Additional problems involved missing imports (e.g., name ‘combinations’ is not defined), logic inconsistencies (e.g., unterminated strings), and glyph-related warnings (Table 7). Beyond code-level issues, many visualizations exhibited low readability, poor chart layout, missing axis labels,

and weak alignment with query intent.

Applying GRPO-based RL significantly mitigated these challenges by leveraging post-execution feedback in a multi-objective reward design. Figure 3 illustrates several representative improvements: (1) syntax and structural errors were eliminated; (2) readability and visual clarity improved through format and visualization rewards; (3) value and logic errors were corrected by enforcing semantic alignment; (4) missing labels were added; (5) charts became query-aligned and visually interpretable; and (6) type errors and invalid operations were resolved.

7 Conclusion

We introduced RL-Text2Vis, the first reinforcement learning framework for text-to-visualization generation that integrates multimodal, post-execution feedback into training. Unlike supervised approaches that optimize token-level likelihoods, RL-Text2Vis jointly optimizes textual accuracy, code executability, and visualization quality through GRPO with a multi-objective reward. Experiments show consistent gains across models and benchmarks, demonstrating that reinforcement learning with post-execution visual feedback enhances structured and multimodal reasoning. The framework matches or surpasses proprietary models like GPT-4o while being open, efficient, and privacy-preserving, highlighting the potential of multi-objective RL for advancing multimodal generation.

Acknowledgments

This work was supported by the Natural Sciences Engineering Research Council (NSERC) of Canada, Canada Foundation for Innovation (CFI), and York University’s Catalyzing Interdisciplinary Research Clusters fund.

Ethical Considerations

RL-Text2Vis is designed to improve transparency in text-to-visualization generation; however, ethical risks remain. Visualizations can amplify biases present in the data or queries, potentially leading to misinterpretation in sensitive domains like healthcare or finance. While our benchmarks use synthetic or public datasets to avoid privacy issues, real-world deployments must ensure data anonymization and human oversight. Additionally, the computational cost of RL raises environmental concerns, which we mitigate through model reuse

and efficient training strategies. Finally, safeguards should be implemented to prevent misuse, such as generating deceptive visualizations or applying the system in adversarial contexts.

Limitations

Although RL-Text2Vis demonstrates significant improvements in text-to-visualization generation, several limitations remain. First, the 14B model yields the best quality but requires considerably more computation and memory. For resource-constrained organizations, the 7B variant is a more practical and cost-effective choice. Second, we did not train larger backbones (e.g., 32B or 72B) due to resource constraints, but our scaling from 3B→7B→14B already shows consistent gains, suggesting that further improvements are likely at larger scales, which remains a promising direction for future work. Third, while our method generalizes to out-of-domain benchmarks, its robustness in highly specialized domains (e.g., medical or financial visualizations) remains untested. Fourth, our study focuses on static visualizations; extending this framework to interactive or multi-view visual analytics is an important direction for future work.

References

- Manuela Aparicio and Carlos J Costa. 2015. Data visualization. *Communication design quarterly review*, 3(1):7–11.
- Nan Chen, Yuge Zhang, Jiahang Xu, Kan Ren, and Yuqing Yang. 2024. Viseval: A benchmark for data visualization in the era of large language models. *IEEE Transactions on Visualization and Computer Graphics*.
- Victor Dibia. 2023. Lida: A tool for automatic generation of grammar-agnostic visualizations and infographics using large language models. *arXiv preprint arXiv:2303.02927*.
- Timur Galimzyanov, Sergey Titov, Yaroslav Golubev, and Egor Bogomolov. 2025. Drawing pandas: A benchmark for llms in generating plotting code. In *2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)*, pages 503–507. IEEE.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

- Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. 2024. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Yucheng Han, Chi Zhang, Xin Chen, Xu Yang, Zhibin Wang, Gang Yu, Bin Fu, and Hanwang Zhang. 2023. Chartllama: A multimodal llm for chart understanding and generation. *arXiv preprint arXiv:2311.16483*.
- Enamul Hoque, Parsa Kavehzadeh, and Ahmed Masry. 2022. Chart question answering: State of the art and future directions. In *Computer Graphics Forum*, volume 41, pages 555–572. Wiley Online Library.
- Mohammed Saidul Islam, Md Tahmid Rahman Laskar, Md Rizwan Parvez, Enamul Hoque, and Shafiq Joty. 2024. [DataNarrative: Automated data-driven storytelling with visualizations and texts](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19253–19286, Miami, Florida, USA. Association for Computational Linguistics.
- Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.
- Shankar Kantharaj, Xuan Long Do, Rixie Tiffany Leong, Jia Qing Tan, Enamul Hoque, and Shafiq Joty. 2022a. Opencqa: Open-ended question answering with charts. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11817–11837.
- Shankar Kantharaj, Rixie Tiffany Leong, Xiang Lin, Ahmed Masry, Megh Thakkar, Enamul Hoque, and Shafiq Joty. 2022b. [Chart-to-text: A large-scale benchmark for chart summarization](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4005–4023, Dublin, Ireland. Association for Computational Linguistics.
- Md Tahmid Rahman Laskar, Sawsan Alqahtani, M Saiful Bari, Mizanur Rahman, Mohammad Abdullah Matin Khan, Haidar Khan, Israt Jahan, Amran Bhuiyan, Chee Wei Tan, Md Rizwan Parvez, et al. 2024. A systematic survey and critical review on evaluating large language models: Challenges, limitations, and recommendations. *arXiv preprint arXiv:2407.04069*.
- Md Tahmid Rahman Laskar, M Saiful Bari, Mizanur Rahman, Md Amran Hossen Bhuiyan, Shafiq Joty, and Jimmy Xiangji Huang. 2023. A systematic study and comprehensive evaluation of chatgpt on benchmark datasets. *arXiv preprint arXiv:2305.18486*.
- Md Tahmid Rahman Laskar, Mohammed Saidul Islam, Ridwan Mahbub, Ahmed Masry, Mizanur Rahman, Amran Bhuiyan, Mir Tafseer Nayeem, Shafiq Joty, Enamul Hoque, and Jimmy Huang. 2025. Judging the judges: Can large vision-language models fairly evaluate chart comprehension and reasoning? *arXiv preprint arXiv:2505.08468*.
- Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. 2022. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:21314–21328.
- Shuaimin Li, Xuanang Chen, Yuanfeng Song, Yunze Song, and Chen Zhang. 2024. Prompt4vis: Prompting large language models with example mining and schema filtering for tabular data visualization. *arXiv preprint arXiv:2402.07909*.
- Can Liu, Yun Han, Ruike Jiang, and Xiaoru Yuan. 2021. Advisor: Automatic visualization answer for natural-language question on tabular data. In *2021 IEEE 14th Pacific Visualization Symposium (PacificVis)*, pages 11–20. IEEE.
- Yuyu Luo, Jiawei Tang, and Guoliang Li. 2021. nvbench: A large-scale synthesized dataset for cross-domain natural language to visualization task. *arXiv preprint arXiv:2112.12926*.
- Paula Maddigan and Teo Susnjak. 2023. Chat2vis: Generating data visualizations via natural language using chatgpt, codex and gpt-3 large language models. *Ieee Access*, 11:45181–45193.
- Tanwi Mallick, Orcun Yildiz, David Lenz, and Tom Peterka. 2024. Chatvis: Automating scientific visualization with a large language model. In *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 49–55. IEEE.
- Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. [ChartQA: A benchmark for question answering about charts with visual and logical reasoning](#). In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2263–2279, Dublin, Ireland. Association for Computational Linguistics.
- Ahmed Masry, Parsa Kavehzadeh, Xuan Long Do, Enamul Hoque, and Shafiq Joty. 2023. [UniChart: A universal vision-language pretrained model for chart comprehension and reasoning](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14662–14684, Singapore. Association for Computational Linguistics.
- Ahmed Masry, Abhay Puri, Masoud Hashemi, Juan A. Rodriguez, Megh Thakkar, Khyati Mahajan, Vikas Yadav, Sathwik Tejaswi Madhusudhan, Alexandre Piché, Dzmitry Bahdanau, Christopher Pal, David Vazquez, Enamul Hoque, Perouz Taslakian, Sai Rajeswar, and Spandana Gella. 2025. [Bigcharts-r1: Enhanced chart reasoning with visual reinforcement](#)

- [finetuning](#). In *Second Conference on Language Modeling*.
- Ahmed Masry, Mehrad Shahmohammadi, Md Rizwan Parvez, Enamul Hoque, and Shafiq Joty. 2024a. [Chartinstruct: Instruction tuning for chart comprehension and reasoning](#).
- Ahmed Masry, Megh Thakkar, Aayush Bajaj, Aaryaman Kartha, Enamul Hoque, and Shafiq Joty. 2024b. [Chartgemma: Visual instruction-tuning for chart reasoning in the wild](#).
- Arpit Narechania, Arjun Srinivasan, and John Stasko. 2020. Nl4dv: A toolkit for generating analytic specifications for data visualization from natural language queries. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):369–379.
- OpenAI. 2024. [Gpt-4 technical report](#).
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.
- Mizanur Rahman, Amran Bhuiyan, Mohammed Saidul Islam, Md Tahmid Rahman Laskar, Ridwan Mahbub, Ahmed Masry, Shafiq Joty, and Enamul Hoque. 2025a. Llm-based data science agents: A survey of capabilities, challenges, and future directions. *arXiv preprint arXiv:2510.04023*.
- Mizanur Rahman, Md Tahmid Rahman Laskar, Shafiq Joty, and Enamul Hoque. 2025b. [Text2vis: A challenging and diverse benchmark for generating multi-modal visualizations from text](#).
- Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Leixian Shen, Enya Shen, Yuyu Luo, Xiaocong Yang, Xuming Hu, Xiongshuai Zhang, Zhiwei Tai, and Jianmin Wang. 2022. Towards natural language interfaces for data visualization: A survey. *IEEE transactions on visualization and computer graphics*, 29(6):3121–3144.
- Arjun Srinivasan, Nikhila Nyapathy, Bongshin Lee, Steven M Drucker, and John Stasko. 2021. Collecting and characterizing natural language utterances for specifying data visualizations. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–10.
- Gemini Team. 2024. [Gemini 1.5: Unlocking multi-modal understanding across millions of tokens of context](#).
- Yuan Tian, Weiwei Cui, Dazhen Deng, Xinjing Yi, Yurun Yang, Haidong Zhang, and Yingcai Wu. 2024. Chartgpt: Leveraging llms to generate charts from abstract natural language. *IEEE Transactions on Visualization and Computer Graphics*, 31(3):1731–1745.
- Guiyao Tie, Zeli Zhao, Dingjie Song, Fuyang Wei, Rong Zhou, Yurou Dai, Wen Yin, Zhejian Yang, Jiangyue Yan, Yao Su, et al. 2025. A survey on post-training of large language models. *arXiv e-prints*, pages arXiv–2503.
- Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2023. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*.
- Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I Wang. 2025. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *arXiv preprint arXiv:2502.18449*.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.
- Chengze Zhang, Changshan Li, and Shiyang Gao. 2025. Mdsf: Context-aware multi-dimensional data storytelling framework based on large language model. *arXiv preprint arXiv:2501.01014*.
- Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. 2024. [Instruction tuning for large language models: A survey](#).

A Appendices

A.1 Reinforcement Learning for LLM Alignment

Reinforcement Learning (RL) provides a framework for optimizing sequential decision-making by learning policies that maximize expected cumulative rewards. Unlike supervised learning, which relies on explicit ground-truth labels, RL agents learn from feedback signals (rewards) derived from interactions with an environment. This paradigm has been successfully applied to robotics, games, and more recently to aligning large language models (LLMs) with human preferences.

A.1.1 Reinforcement Learning from Human Feedback (RLHF)

RLHF (Ouyang et al., 2022) aligns LLMs with human preferences through three steps: (i) *Supervised Fine-Tuning (SFT)* on curated demonstrations, (ii) training a *reward model* from human comparisons, and (iii) optimizing the policy using a reinforcement learning algorithm, typically Proximal Policy Optimization (PPO). While highly effective, RLHF is computationally expensive.

A.1.2 Proximal Policy Optimization (PPO)

PPO (Schulman et al., 2017) is the foundation of RLHF and provides stable policy updates via a clipped surrogate objective:

$$\mathcal{L}_{\text{PPO}} = \mathbb{E} \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right], \quad (4)$$

where $r_t(\theta)$ is the probability ratio between new and old policies, and \hat{A}_t is the advantage function. PPO employs a value function (critic) for advantage estimation and uses KL regularization for stable updates.

A.1.3 Direct Preference Optimization (DPO)

DPO eliminates the need for a reward model by optimizing the policy directly on human preference data. Given a preferred response y^+ and a dis-preferred response y^- for the same input x , DPO maximizes:

$$\mathcal{L}_{\text{DPO}} = -\log \sigma \left(\beta \left[\log \frac{\pi_{\theta}(y^+|x)}{\pi_{\text{ref}}(y^+|x)} - \log \frac{\pi_{\theta}(y^-|x)}{\pi_{\text{ref}}(y^-|x)} \right] \right), \quad (5)$$

where β controls the sharpness of preference alignment. Unlike PPO, DPO skips the RL loop and directly uses preference pairs to guide updates.

A.1.4 Group Relative Policy Optimization (GRPO)

GRPO (Shao et al., 2024) extends PPO for settings where explicit value functions are impractical. Instead of using a critic, GRPO estimates advantages via *within-group reward standardization*. For each input, the model generates a group of G candidate outputs, assigns a scalar reward r_i to each, and computes the normalized advantage:

$$\hat{A}_i = \frac{r_i - \bar{r}}{\sigma_r}, \quad (6)$$

where \bar{r} and σ_r are the group mean and standard deviation.

The GRPO objective follows a PPO-style clipped surrogate with KL regularization:

$$\mathcal{L}_{\text{GRPO}} = \mathbb{E} \left[\min \left(r_i(\theta) \hat{A}_i, \text{clip}(r_i(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_i \right) - \beta D_{\text{KL}}(\pi_{\theta} \| \pi_{\text{ref}}) \right]. \quad (7)$$

GRPO is well-suited for tasks like RL-Text2Vis, where key metrics—textual correctness, code executability, and visualization clarity—are only observable post-execution.

A.1.5 Key Differences: DPO vs. GRPO

Both methods optimize from preference signals, but DPO operates on pairwise preferences without an RL loop, while GRPO generalizes to groups and retains PPO-like stability. It is more suitable for tasks requiring structured outputs and multimodal optimization.

A.2 Additional Implementation Details

A.2.1 In-loop evaluators

To match real-world deployment and privacy constraints, we prioritized open-source evaluators for in-loop rewards. For the visual reward, we first compared Qwen2.5-VL-3B with LLaMA-3.2-3B VLM on a 100-sample set and found that Qwen achieved better agreement with human ratings on visualization quality. Based on this, we adopted Qwen2.5-VL-3B (and its larger variants) as the in-loop visual judge. For text rewards, we used Qwen2.5-7B for the 7B policy and Qwen2.5-14B for the 14B policy; for visual rewards, we used Qwen2.5-VL-7B and Qwen2.5-VL-32B, respectively. To mitigate evaluator bias, we conducted a stratified 200-sample comparison of rewards from Qwen2.5-7B, Qwen2.5-VL-7B, and GPT-4o, and observed strong cross-judge agreement (Pearson

Configuration	Code Exec. Success (%)	Answer Match (%)	Visual Clarity Readability	Chart Correctness	Final Pass Rate (%)
RL-Text2Vis-7B (Gen-4)	87	28	3.62	3.52	17
RL-Text2Vis-7B (Gen-8)	91	31	3.84	3.86	22

Table 5: Impact of GRPO group size on Qwen2.5-7B.

$r = 0.85\text{--}0.97$), indicating that the policy did not overfit to any single evaluator’s biases.

A.2.2 Hardware and optimization

Training was conducted on high-performance hardware: Qwen2.5-7B-Instruct was fine-tuned on $4\times$ NVIDIA A100 80 GB GPUs, and Qwen2.5-14B-Instruct on $6\times$ NVIDIA H100 80 GB GPUs, both for two epochs. We used AdamW (learning rate 1×10^{-5} , weight decay 0.1) with a cosine scheduler, KL penalty $\beta = 0.04$, and gradient-norm clipping $= 0.1$. Gradient checkpointing and `bf16` mixed precision were enabled to reduce memory usage. These hyperparameters were selected via a small grid search, choosing the configuration that maximized performance on a fixed 10% development subset of `Text2Vistest1`. The full training process required approximately 25 hours for the 7B model and 50 hours for the 14B model.

A.3 Error Analysis

Figure 3 presents a qualitative comparison between the zero-shot Qwen2.5-14B model and the RL-Text2Vis-14B model. The analysis highlights that GRPO significantly improves visualization quality by reducing common issues such as non-executable code, misaligned charts, and low readability. Compared to baselines, our RL-trained models produce outputs that are more interpretable and semantically aligned with the query.

A.4 Training Dynamics

We visualize sample training dynamics over 150 RL steps for RL-Text2Vis-14B. Each figure highlights different metrics tracked during GRPO training.

A.5 Prompt Templates for Evaluation Metrics

For all automated evaluations, we followed the official prompt templates provided in the Text2Vis benchmark (Rahman et al., 2025b). These templates define instructions for assessing answer correctness, code executability, and visualization quality across readability and correctness dimensions.



Figure 3: Error analysis before and after GRPO. GRPO significantly improves text-to-visualization generation by resolving errors such as syntax, value errors, enhancing readability, visual quality, and alignment with the query.

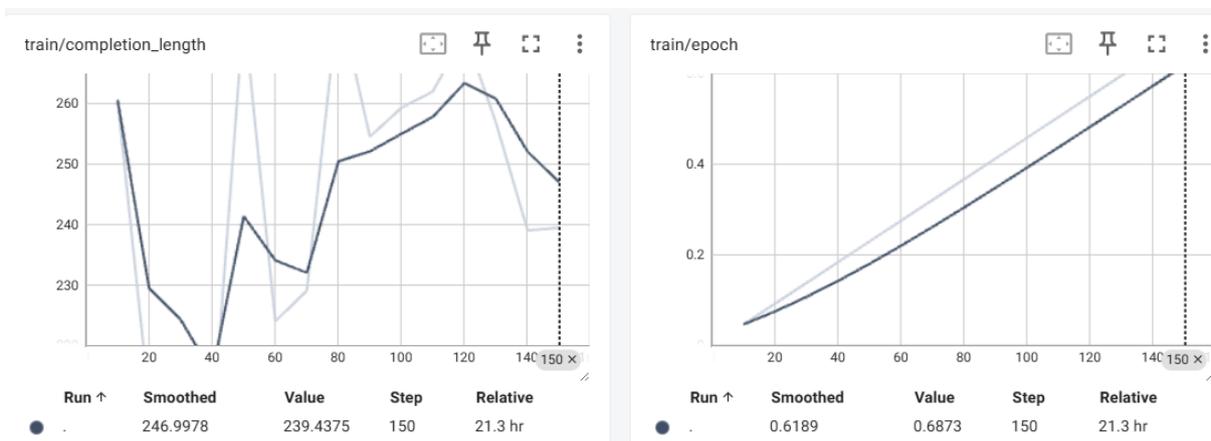


Figure 4: Completion length and epoch progression. The model stabilizes in output length while training steps progress linearly.

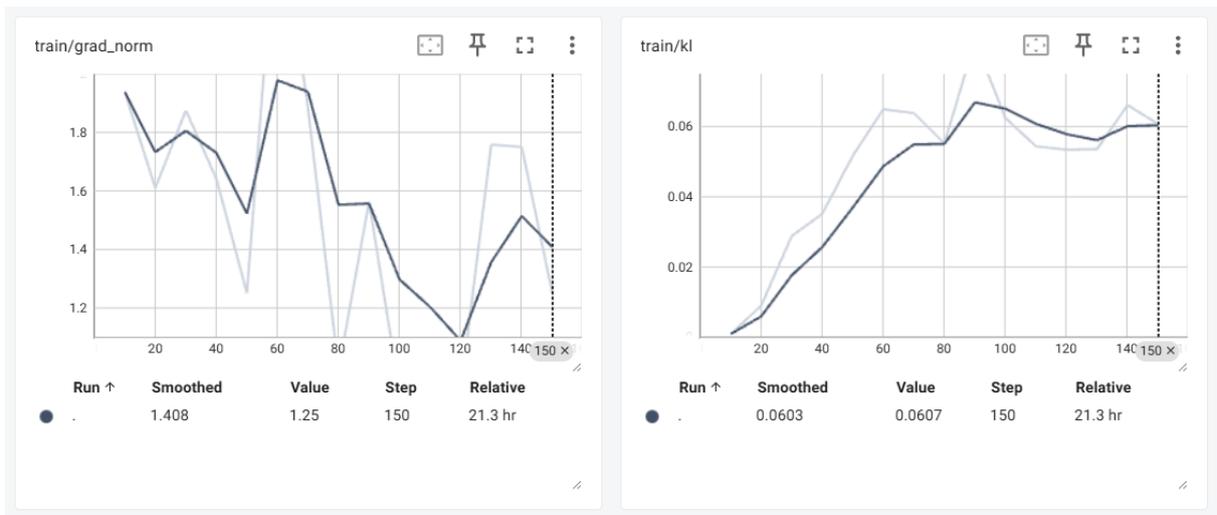


Figure 5: Gradient norm and KL divergence. Indicates optimization stability and policy deviation from the reference model.

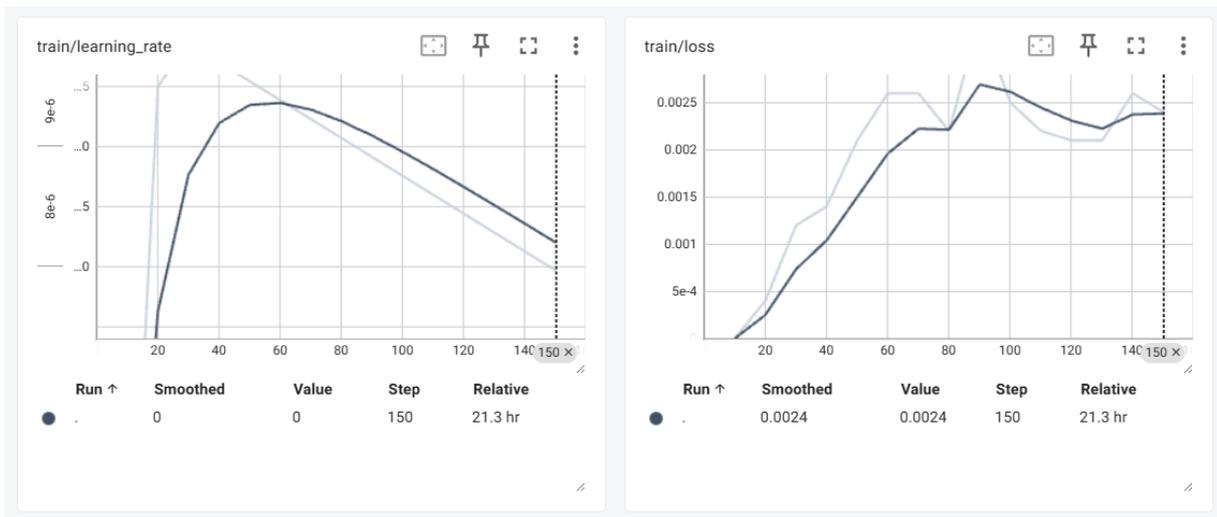


Figure 6: Learning rate schedule and training loss. Warm-up and decay patterns are followed, with loss trends influenced by reward-maximizing objectives.

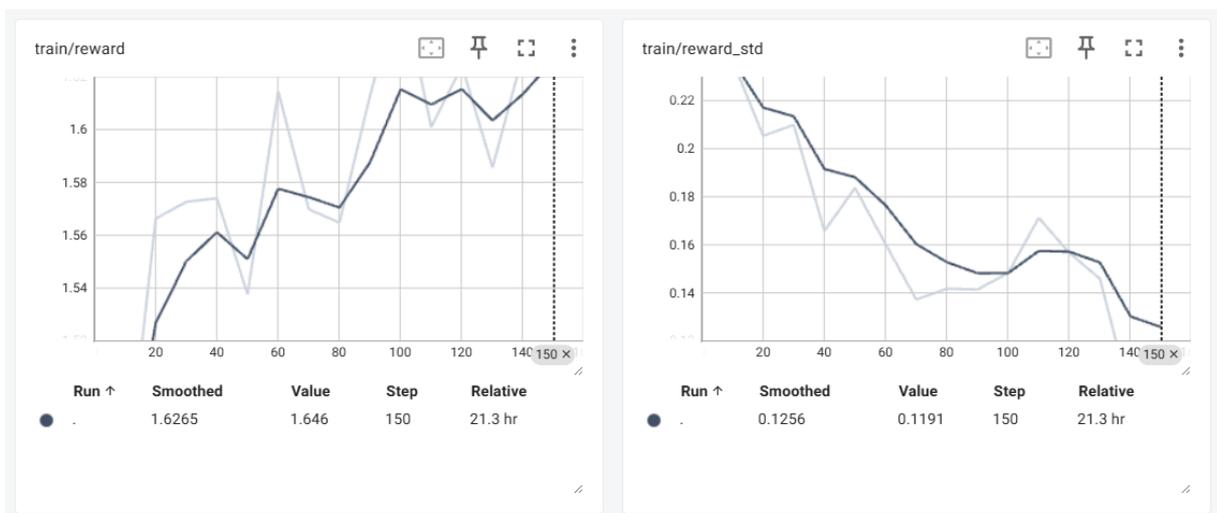


Figure 7: Average reward and reward standard deviation. Reflects growing reward consistency and reduced variance across outputs.

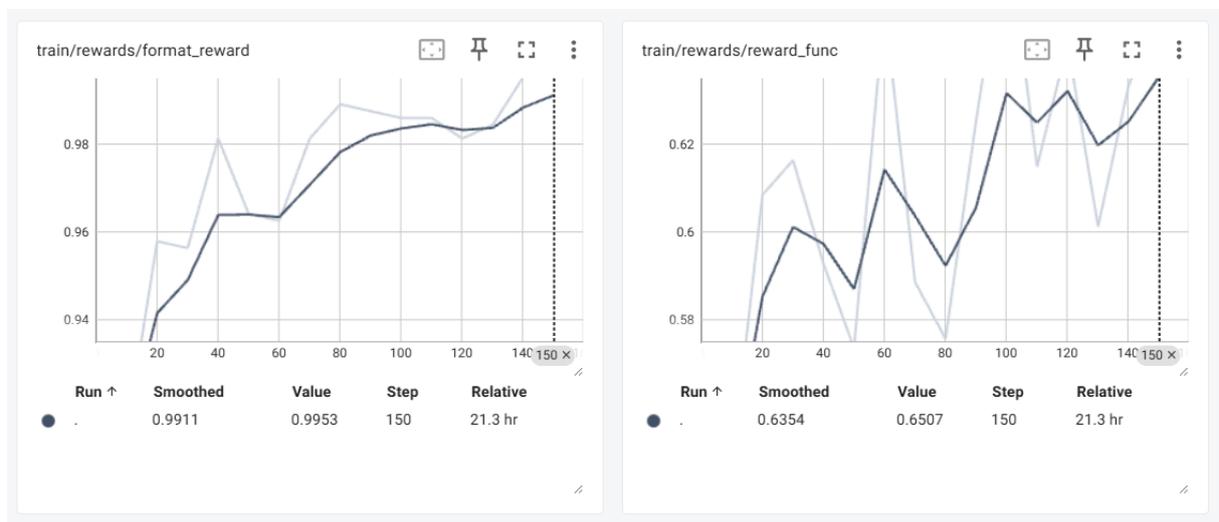


Figure 8: Format reward and Composite reward function score. Tracks alignment with structural and multimodal feedback objectives.

Category	Prompt Template
Evaluation	<p>You are an evaluation expert responsible for assessing the accuracy of generated answers and the quality of visualizations. Given a structured data table, a user-generated question, a model-generated response, and an image-based visualization, your task is to validate the correctness of the response and evaluate the visualization quality.</p> <p>Input Data:</p> <ul style="list-style-type: none"> • Data Table: {row['Table Data']} • Question: {row['Generated Question']} • Generated Answer: {row['Generated Answer']} • Ground Truth Answer: {row['Answer']} • Generated Image: {row['Generated image']} <p>Task:</p> <ol style="list-style-type: none"> 1. Answer Matching: Compare the generated answer with the ground truth using following evaluation criteria. 2. Visualization Evaluation: Score the visualization based on following evaluation criteria. <p>Evaluation Criteria:</p> <ol style="list-style-type: none"> 1. Answer Matching (Binary: 1 or 0) <ul style="list-style-type: none"> • Match if numbers are close (e.g., "48.77" vs "48.73") or equivalent percentage formats (e.g., "100" vs "100") • Match if the ground truth appears within the generated response (e.g., "100" in "The result is 100"). • For long ground truth answer, match is considered as long as the core summary remains the same, even if the wording differs. • Allow minor spelling variations or abbreviations (e.g., "Albenia" vs "Albania", "USA" vs "United States"). • No match if the meaning changes significantly (e.g., "Fragile" vs "Extreme fragility"). 2. Readability and Quality Score (0-5) <ul style="list-style-type: none"> • Labels and Titles: Are they clear, concise, and correctly positioned? • Layout Spacing: Is the layout well-organized with no clutter? • Color Accessibility: Are colors distinct and accessible (colorblind-friendly)? • Axis Scaling: Are axes correctly labeled and proportional? • Chart Type Suitability: Is the visualization appropriate for the data type (e.g., line chart for trends)? • Font and Legends: Are fonts readable, and legends properly aligned? • Annotation Readability: Are annotations (e.g., data labels, callouts) clear, well-placed, and non-overlapping? 3. Chart Correctness Score (0-5) <ul style="list-style-type: none"> • Query Alignment: Does the visualization correctly address the question? • Data Integrity: Are all data points accurately plotted? • Insight Representation: Does the chart effectively communicate its key insights based on its type? • Handling Missing Data: Is missing data presented appropriately without misleading distortion? • Complexity Handling: For multi-step queries, is the visualization logically structured? <p>• 5.0 – Excellent: Clear, accurate, and no issues.</p> <p>• 4.5 – Very Good: Minor issues but does not impact understanding.</p> <p>• 4.0 – Good: Small flaws like minor misalignments.</p> <p>• 3.5 – Decent: Some readability/accuracy issues but still interpretable.</p> <p>• 3.0 – Average: Noticeable problems that affect clarity or correctness.</p> <p>• 2.5 – Below Average: Several issues that may lead to misinterpretation.</p> <p>• 2.0 – Poor: Significant issues making the chart unclear.</p> <p>• 1.5 – Very Poor: Major readability or correctness flaws.</p> <p>• 1.0 – Unusable: Completely unclear or misleading.</p> <p>• 0.0 – Failed: The visualization is unreadable or irrelevant.</p> <p>Output Requirements:</p> <ul style="list-style-type: none"> • Ensure the final output is in a valid JSON format without additional text. <p>Expected JSON Output Format:</p> <pre>{ "Answer Match": "...", "Readability and Quality Score": "...", "Chart Correctness Score": "..." }</pre>

Table 6: Prompt Template for Evaluating Results Using the GPT-4.o Model (Rahman et al., 2025b)

Error Type	Qwen2.5-7B	Qwen2.5-14B	RL-Text2Vis-7B	RL-Text2Vis-14B	GPT-4o	Gemini-1.5 Flash	Llama-3.1-8B	CodeLlama-7B
TypeError	5.86	0.84	1.67	–	7.62	4.03	7.62	25.54
ValueError	5.02	1.67	1.26	1.67	1.61	1.97	2.92	3.28
SyntaxError	4.60	15.48	1.67	–	3.33	7.11	9.67	2.52
NameError	2.09	0.42	1.26	0.54	0.20	0.71	7.66	7.03
KeyError	1.67	–	0.84	–	0.15	0.25	0.40	–
IndexError	0.42	–	–	–	0.40	0.20	0.45	–
OtherError	3.10	–	2.10	0.42	0.60	2.10	0.74	0.35

Table 7: Error-type distribution (%) across models on Text2Vis.