CharTide: Data-Centric Chart-to-Code Generation via Tri-Perspective Tuning and Inquiry-Driven Evolution

Xiangxi Zheng[1], Kuang He[2], Jiayi Hu[3], Ping Yu[1], Rui Yan[4], Yuan Yao[1†], Peng Hou[2], Anxiang Zeng[2], Alex Jinpeng Wang[5†],

[1]Nanjing University [2]LLM Team, Shopee Pte. Ltd. [3]East China Normal University
[4]Nanjing University of Science and Technology [5]Central South University
{zhengxx, y.yao}@nju.edu.cn, jinpengwang@csu.edu.cn

Abstract

Chart-to-code generation demands strict visual precision and syntactic correctness from Vision-Language Models (VLMs). However, existing approaches are fundamentally constrained by **data-centric limitations**: despite the availability of growing chart-to-code datasets, simply scaling homogeneous chart-code pairs conflates visual perception with program logic, preventing models from fully leveraging the richness of multimodal supervision. We present CharTide, a novel **data-centric framework** that systematically redesigns both training and alignment data for chart-to-code generation. First, we construct a **2M-sample** dataset via a **Tri-Perspective Tuning** strategy, explicitly decoupling training into visual perception, pure-text code logic, and modality fusion streams, enabling a 7B model to surpass specialized baselines using only supervised data. Second, we reformulate alignment as a **data verification** problem rather than a heuristic scoring task. To this end, we introduce an **Inquiry-Driven RL** framework grounded in the principle of information invariance: a downstream model should yield consistent answers to identical visual queries across both original and generated charts. Moving beyond rigid rule matching or VLM scoring, we employ a frozen Inspector to objectively verify generated charts through atomic QA tasks, providing verifiable reward signals based on answer accuracy. Experiments on ChartMimic, Plot2Code, and ChartX show that **CharTide-7B/8B** significantly outperforms open-source baselines, surpasses GPT-4o, and is competitive with GPT-5.

1 Introduction

Recent advancements in Multimodal Large Language Models (MLLMs) have revolutionized vision-language understanding (Bai et al., 2025a,b; Wang et al., 2025; Masry et al., 2022, 2025), and have further extended to multimodal code generation for UIs and sketches (Si et al., 2025; Jiang et al., 2025b; Wan et al., 2025; Yun et al., 2024; Gui et al., 2025). However, applying such capabilities to **Chart-to-Code generation** presents a unique challenge stemming from the fundamental difference in data characteristics. Unlike open-ended chart understanding (Kondic et al., 2025; He et al., 2024; Xing et al., 2025; Chen et al., 2025b), chart-to-code data requires modeling the tight coupling between continuous numerical values and precise visual styling. This task requires reverse-engineering these dense visual signals into rigorous plotting code under zero-tolerance constraints on both visual precision and syntactic correctness. Consequently, standard data paradigms effective in general domains fall short here, as they demand that models simultaneously possess fine-grained perception and precise code synthesis capabilities (Yang et al., 2024; Xia et al., 2025) to handle such strict information mapping.

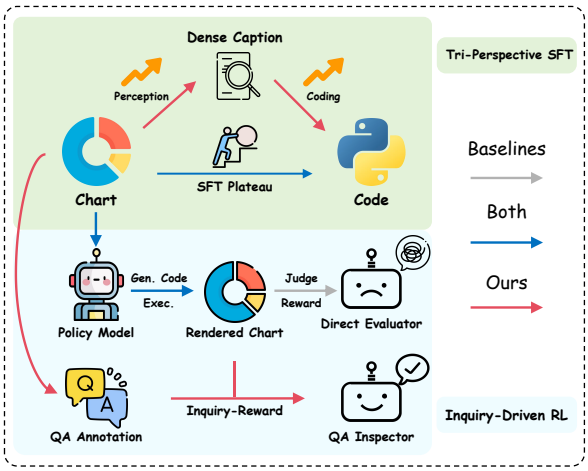

Figure 1: **Overview of the Data-Centric CharTide Framework.** (Top) **Tri-Perspective SFT** explicitly decouples data streams to break the performance plateau. (Bottom) **Inquiry-Driven RL** replaces subjective scoring with objective, fact-based verification.

[†]Corresponding authors: Yuan Yao, Alex Jinpeng Wang

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 24966–24985

While recent efforts have shifted to real-world data, the Supervised Fine-Tuning (SFT) paradigm has hit a distinct **scaling wall**. Empirical studies such as MSRL (Chen et al., 2025a), which scales to 3M samples, indicate that merely accumulating homogeneous data (i.e., Chart-Code pairs) yields diminishing returns. We attribute this bottleneck not only to data volume but also to the *inherent inefficiency* of the singular Chart-Code pair format. Fundamentally, plotting code is a highly structured modality in which boilerplate syntax and non-visual logic occupy a disproportionate share of the token space, thereby diluting supervision for key visual attributes. This **information asymmetry** biases models towards **template memorization** during end-to-end training, leading them to prioritize syntactic patterns rather than achieving deep alignment with fine-grained visual perception.

A similar insufficiency in supervision signals extends to the Reinforcement Learning (RL) stage. Due to the absence of precise and verifiable evaluation mechanisms, existing approaches rely on subjective, black-box VLM scoring (Chen et al., 2025a) or rigid rule-based matching (Tan et al., 2025). Consequently, they fail to **leverage** the **diverse high-level semantics** embedded in high-quality data, resulting in supervision that is unstable, difficult to reproduce, and data-inefficient.

To address these challenges, we propose Data-Centric Chart-to-Code Generation via **T**ri-Perspective Tuning and **I**nquiry-**D**riven **E**volution (**CharTide**). First, we introduce a **Tri-Perspective Decomposed SFT** strategy to overcome the scaling wall. Recognizing that merely scaling data fails to disentangle visual and logical hallucinations (Chen et al., 2025a), we construct orthogonal data streams that decouple training into fine-grained perception, pure-text code logic, and modality fusion. This decomposition effectively breaks the homogeneity bottleneck, enabling our 7B-scale model to distill capabilities that surpass 235B-scale baselines on ChartMimic (Yang et al., 2024).

Furthermore, we propose **Chart2Code Inquiry-Driven RL**, shifting the alignment paradigm from subjective preference to verifiable fact-checking. Grounded in the hypothesis that accurate reproduction ensures the fidelity of information transfer, we replace unstable black-box VLM scoring (Chen et al., 2025a; Zhang et al., 2025b) with a deterministic Inspector. This mechanism objectively verifies generated charts via rigorous QA pairs and visual constraints, ensuring the generated code is not just semantically precise in data trends but also maintains high pixel-level fidelity.

Experiments on three authoritative benchmarks, including ChartMimic (Yang et al., 2024), Plot2Code (Wu et al., 2025), and ChartX (Xia et al., 2025), demonstrate that CharTide achieves state-of-the-art (SOTA) performance among open-source models, surpassing GPT-4o and achieving performance comparable to GPT-5.

Our main contributions are as follows:

- We propose Tri-Perspective Decomposed SFT to address data homogeneity by decoupling visual perception, code logic, and modality fusion. A curated 2M-sample dataset enables 7B models to acquire complementary chart-to-code capabilities and outperform 235B-scale baselines.
- We integrate Inquiry-Driven verifiable rewards by reformulating alignment as data verification. By leveraging a frozen Inspector to objectively verify outputs via curated QA data, our approach substantially improves semantic consistency and pixel-level fidelity of the generated code.
- We present **CharTide**, achieving SOTA performance among open-source models and demonstrating competitive performance with GPT-5. Extensive experiments validate the efficacy of our two-stage pipeline, offering robust insights for future advancements in this domain.

2 Related Work

2.1 Chart-to-Code Generation

The Chart-to-Code task demands pixel-level visual reproduction through executable code, imposing dual constraints on visual perception and syntactic correctness (Yang et al., 2024; Wu et al., 2025; Xia et al., 2025). Representative chart-to-code methods (Han et al., 2023; Zhao et al., 2025b) mainly formulate the task as end-to-end supervised generation from charts to plotting code, typically using synthetic or automatically constructed chart–code pairs. However, synthetic datasets inherently suffer from limited stylistic diversity. While recent works (Tan et al., 2025; Zhao et al., 2025a; Chen et al., 2025a; Niu et al., 2025; Jiang et al., 2025a) have pivoted to real-world annotations to improve diversity, concurrent research (Chen et al., 2025a) identifies a scaling wall in the SFT paradigm: mere data accumulation without capability decoupling fails to address the entanglement of visual and logical hallucinations.

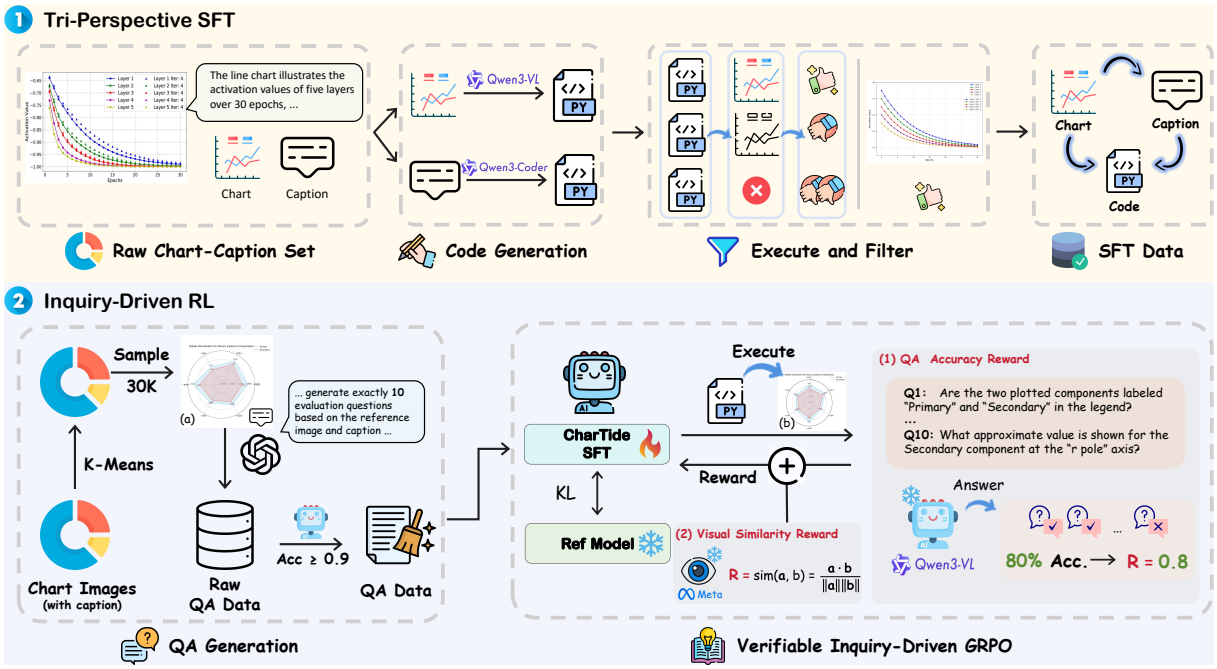

Figure 2: **The detailed pipeline of CharTide.** The pipeline consists of two stages: (1) **Tri-Perspective SFT** constructs three complementary data streams, including Visual Perception, Code Logic, and Modality Fusion to distill multi-dimensional capabilities into the foundational model; (2) **Inquiry-Driven RL** aligns the model using a hybrid verification loop, where a frozen Inspector provides objective semantic rewards (r_{QA}) via fact-checking, complemented by visual similarity rewards (r_{vis}) to ensure rigorous pixel-level and semantic alignment.

To overcome the one-to-many mapping ambiguity in code generation, RL has been widely adopted (Zhang et al., 2025b; Tan et al., 2025; Chen et al., 2025a). Existing RL strategies fall into two categories, yet both face significant limitations:
(1) **Rule-based methods** (Tan et al., 2025; Zhang et al., 2025b) rely on heuristic attribute matching (e.g., colors, legends), which neglects holistic visual semantics and generalizability;
(2) **VLM-as-a-Judge methods** (Chen et al., 2025a) utilize visual similarity scores from large VLMs. However, these approaches struggle with the subjective, black-box nature of VLM scoring, leading to high variance and computational costs. In contrast, CharTide introduces an objective, verification-based reward mechanism to resolve these issues.

2.2 Reinforcement Learning for MLLMs

The success of reasoning models such as DeepSeek-R1 (DeepSeek-AI, 2025) has ignited significant interest within the multimodal domain regarding Reinforcement Learning with Verifiable Rewards (RLVR) (Zhang et al., 2025a; Lambert et al., 2024; Chen et al., 2025c; Zha et al., 2025; Chen et al., 2025b). Unlike traditional alignment strategies based on human preferences, RLVR optimizes policies leveraging objective and deterministic outcomes. Pioneering works (Huang et al., 2025; Meng et al., 2025; Yang et al., 2025b) utilize GRPO combined with accuracy-based rewards, successfully fostering emergent capabilities in multimodal reasoning. In other adaptations, CapRL (Xing et al., 2025) pioneered the use of downstream task QA performance to implement verifiable rewards for image captioning models, surpassing simple text matching metrics. Inspired by this, researchers have explored various reward forms for different tasks (Xing et al., 2025; Ni et al., 2025), offering new directions for future research.

3 Method

We present **CharTide**, a data-centric framework for precise Chart-to-Code generation. As illustrated in Figure 2, the framework initially establishes a robust foundation via **Tri-Perspective Decomposed SFT**, followed by semantic and pixel-level data verification through **Inquiry-Driven RL**.

3.1 Tri-Perspective Decomposed SFT

To overcome the bottlenecks caused by data homogeneity, we leverage charts and high-quality captions from ChartCap (Lim et al., 2025) as our core data source. Via multi-view distillation, we construct three **complementary** training streams, each

targeting a distinct capability dimension:

Visual Perception Stream (Chart → Caption). To bridge the perception gap inherent in 7B-scale models, we focus on strictly aligning visual features with dense textual descriptions from ChartCap. We apply a simple length-based filter to exclude excessively long captions, ensuring the model focuses on concise and effective visual grounding without context overflow.

Code Logic Stream (Caption → Code). To decouple syntax learning from visual perception, we construct a pure-text code generation stream. We prompt Qwen3-Coder-30B-A3B (Yang et al., 2025a) with detailed captions to generate plotting code that strictly reflects the textual descriptions. To ensure quality, we execute and filter the outputs through a visual consistency check using Qwen3-VL-235B-A22B (Bai et al., 2025a) given the caption and original chart. This ensures that we retain only high-quality samples to strengthen the model's logic proficiency without visual interference.

Modality Fusion Stream (Chart → Code). To establish end-to-end generation capability, we integrate 500k samples from ChartCap with publicly available chart datasets (Zhao et al., 2025b,a), totaling 1M chart images. We utilize Qwen3-VL-235B-A22B to generate plotting code from the source charts, filtering the outputs based on WebSSL feature similarity. Instead of pairing the code with the original source images, we strictly pair the generated code with its corresponding rendered image. This strategy eliminates potential discrepancies between the visual chart and the code, ensuring strict pixel-level correspondence and mitigating noise from imperfect ground-truth code.

Ultimately, we combine these three streams with open-source instruction data to construct $\mathcal{D}_{SFT}$ ($\approx$ 2M samples), performing full-parameter fine-tuning on Qwen2.5-VL-7B-Instruct and Qwen3-VL-8B-Instruct. The detailed data construction pipeline is provided in Appendix B.

3.2 Inquiry-Driven RL

Although SFT endows the model with foundational capabilities, it struggles to handle long-tail details. The teacher-forcing paradigm restricts the model's exploration capabilities and output diversity within the code space. Traditional RL approaches face a dilemma: rule-based matching relies on rigid templates, while VLM-as-a-Judge methods suffer from subjective black-box noise. To provide reliable and verifiable supervision, we propose a hybrid verification loop, as illustrated in Figure 2.

VQA Data Preparation. To provide high-quality supervisory signals, we constructed a specialized Chart-VQA dataset. First, we performed K-Means clustering on 500k ChartCap samples using WebSSL-1B features to select 30k representative charts. We then employed GPT-5 to generate a diverse set of QA pairs ($N = 10$ per image), covering dimensions such as titles, legends, and numerical trends, with explicit tolerance annotations for numerical values. To mitigate the inherent hallucination noise of the Inspector, we implemented consistency filtering prior to RL: we utilized Qwen3-VL-30B-A3B to pre-screen the VQA data, retaining only chart samples where the model successfully answers at least 9 questions correctly (Acc ≥ 0.9). This ensures the objectivity of the reward signal by validating the Inspector's understanding of the ground truth. Ultimately, we retained approximately 20k images and high-quality QA data for reinforcement learning, denoted as $\mathcal{D}_{RL} = \{(I_{src}, \mathcal{Q})\}$, where $\mathcal{Q} = \{(q_i, a_i)\}_{i=1}^{N}$. Further details regarding the data construction pipeline are provided in Appendix C.

Inquiry-Driven Rewards. During training, we sample $(I_{src}, \mathcal{Q}) \sim \mathcal{D}_{RL}$. The policy π_θ generates code to render a predicted image I_{pred}. We define the semantic consistency reward r_{QA} as the pass rate verified by the Inspector:

$$r_{QA} = \frac{1}{|\mathcal{Q}|} \sum_{(q,a)\in\mathcal{Q}} \mathbb{I}\left(\mathcal{M}(\text{Inspector}(I_{pred}, q), a)\right) \tag{1}$$

where $\mathbb{I}(\cdot)$ is the indicator function and $\mathcal{M}$ checks semantic alignment incorporating numerical tolerance. While r_{QA} ensures semantic accuracy, the "One-to-Many" nature of code generation may lead to visually suboptimal styling. To impose structural constraints, we introduce a visual consistency reward r_{vis} based on WebSSL-1B, which we empirically found superior to DINO or SigLIP in detecting structural collapse:

$$r_{vis} = \text{CosineSim}(\text{Enc}_{web}(I_{src}), \text{Enc}_{web}(I_{pred})) \tag{2}$$

The total reward is computed as $R_{total} = r_{QA} + \lambda \cdot r_{vis}$, where λ is a balancing coefficient. We optimize the policy π_θ using Group Relative Policy Optimization (GRPO) (Shao et al., 2024). For each query, we sample a group of outputs $\{o_i\}_{i=1}^{G}$ and compute the advantage $\hat{A}_i$ by standardizing the

Table 1: Comparison with Closed-Source and Open-Source Models on ChartMimic, Plot2Code, and ChartX. *For Plot2Code, we change the evaluator from GPT-4V to GPT-4o to ensure robust and reproducible scoring. We report normalized scores over the full test set to avoid survivorship bias. Details are provided in **Appendix A**.

Model	ChartMimic			Plot2Code*			ChartX
	Exec Rate	Low Level	High Level	Exec Rate	Text Match	Rating	GPT score
Full Score	*100*	*100*	*100*	*100*	*100*	*10*	*5*
Proprietary Models							
GPT-4o	94.7	80.0	87.7	87.1	52.6	5.66	2.61
GPT-5	96.8	82.1	94.7	87.8	61.9	7.28	3.59
Gemini-2.5-Pro	94.7	79.2	92.5	88.6	69.1	7.45	3.27
Open-Source General-Domain							
Qwen2.5-VL-7B	75.0	49.0	51.8	68.9	33.7	3.04	2.74
Qwen2.5-VL-72B	75.3	51.9	56.6	59.3	33.2	3.61	2.85
Qwen3-VL-8B	81.7	63.7	71.5	76.5	36.3	3.91	2.93
Qwen3-VL-30B-A3B	85.2	67.7	76.5	87.1	46.7	4.85	2.73
Qwen3-VL-235B-A22B	93.3	76.8	87.6	84.8	46.2	5.19	**3.35**
Open-Source Chart-Domain							
ChartCoder-7B	89.5	72.1	78.5	68.9	31.1	2.73	2.79
ChartMaster-7B	93.5	77.1	83.3	89.4	53.6	4.73	2.82
MSRL-7B-SFT	92.6	71.2	82.8	77.3	34.7	3.71	3.19
MSRL-7B	94.3	76.1	87.4	62.9	31.9	3.24	3.22
VinciCoder-7B	91.2	77.0	83.4	68.9	33.6	3.39	3.18
VinciCoder-8B	90.2	75.8	81.4	85.6	49.8	4.49	3.21
CharTide-7B-SFT	94.3	79.3	86.4	88.6	58.2	5.17	3.00
CharTide-7B	96.7	81.7	91.6	89.4	59.6	5.60	3.22
CharTide-8B-SFT	93.7	80.9	89.4	86.4	58.1	5.46	3.19
CharTide-8B	**97.3**	**83.0**	**92.7**	**91.7**	**64.6**	**5.93**	3.23

total rewards within the group. The model is then updated to maximize this group-relative advantage, incorporating a KL divergence penalty to constrain the policy updates within a stable trust region.

In contrast to VLM-as-a-Judge paradigms that suffer from subjective holistic scoring, our approach simplifies evaluation into atomic verification tasks. By decoupling the evaluator's perception from the generator's performance via ground-truth pre-filtering, we effectively transform stochastic black-box assessments into deterministic, low-variance supervision signals.

4 Experiments

4.1 Implementation Details

We build CharTide-7B and 8B on Qwen2.5-VL-7B and Qwen3-VL-8B backbones, respectively.

SFT Stage: We perform full-parameter fine-tuning on the 2M multi-perspective dataset. We set the global batch size to 256, the initial learning rate to $1e{-5}$. This process requires approximately 36 hours on 8 $\times$ H100 GPUs.

RL Stage: We initialize the policy with the SFT checkpoint and optimize it using the $\approx$ **20k** verified VQA samples. We set the global batch size to 128, the learning rate to $1e{-6}$, and the KL penalty coefficient $\beta = 0.02$. Training is conducted on 8 $\times$ H100 GPUs for the policy model, with an additional 4 $\times$ H100 GPUs allocated for the inference of the frozen Inspector and WebSSL reward models. The RL training completes in roughly 20 hours.

4.2 Main Results

We evaluate CharTide against state-of-the-art proprietary and open-source models on ChartMimic (Yang et al., 2024), Plot2Code (Wu et al., 2025), and ChartX (Xia et al., 2025). As shown in Table 1, CharTide achieves SOTA performance, surpassing both large general VLMs and special-

Table 2: **Ablation of SFT Data Strategies.** We verify the impact of incrementally adding data sources. **C2C**: Chart-to-Code; **Cap**: Chart-to-Caption ; **Cap2C**: Caption-to-Code. Incremental integration of decoupled training streams yields consistent performance gains.

Data Source			ChartMimic		
C2C	Cap	Cap2C	Exec R.	Low L.	High L.
400K	-	-	84.5	68.9	75.7
800K	-	-	91.3	77.5	85.3
1M	-	-	92.0	77.6	85.1
1M	500K	-	92.5	78.8	86.4
1M	500K	400K	**94.3**	**79.3**	**87.4**

ized chart-to-code models, including MSRL (Chen et al., 2025a) and ChartMaster (Tan et al., 2025). On ChartMimic, CharTide-7B attains a High-Level score of **91.6**, outperforming the previous best open-source model MSRL-7B (87.4) and **GPT-4o** (87.7). This trend holds on Plot2Code and ChartX, where CharTide consistently leads open-source models and remains competitive with top-tier proprietary systems such as **GPT-5**. Overall, these results demonstrate that CharTide effectively bridges the capability gap between open-weights and top-tier closed-source models.

4.3 Ablation Study

4.3.1 SFT and RL Ablation

We conduct comprehensive ablation studies on ChartMimic to evaluate the contribution of each component within CharTide.

SFT Strategy Ablation. We evaluate the effectiveness of the Tri-Perspective decoupled data strategy in Table 2. Rows 1-3 show that scaling homogeneous Chart-to-Code (C2C) data quickly saturates, with marginal gains when increasing data from 800K to 1M. To overcome this bottleneck, we progressively introduce decoupled data streams. Adding 500k Image-Caption data (Row 4) improves Low/High-level scores while preserving execution rate, evidencing an enhancement in fine-grained visual perception. Further incorporating 400k Caption-to-Code data (Row 5) markedly increases the **Execution Rate (92.5 $\rightarrow$ 94.3)**, confirming that pure-text logic training effectively isolates syntax learning from visual noise. Overall, the complementary streams jointly contribute to performance gains across multiple dimensions.

Impact of SFT Basis and RL Universality. Table 3 illustrates the symbiosis between SFT and RL. We observe that directly applying RL to the raw

Table 3: **Impact of SFT Basis and RL Universality.** We validate the effectiveness of RL on different foundations, including third-party models (ChartMaster) and different training stages.

Base Model	Stage		ChartMimic		
	SFT	RL	Exec R.	Low L.	High L.
ChartMaster-7B			93.5	77.1	83.3
		✓	96.0	79.2	84.6
Qwen2.5-VL-7B			75.0	49.0	51.8
		✓	94.5	68.0	76.7
	✓		94.3	79.3	86.4
	✓	✓	**96.7**	**81.7**	**91.6**

Table 4: **Ablation of Visual Encoders and Reward Mechanisms.** Section (a) compares visual backbones using only R_{vis}. Section (b) fixes the backbone to WebSSL-1B and the inspector to Qwen3-VL-30B-A3B to verify the effectiveness of our Inquiry-Driven strategy against VLM-Judge baselines.

Settings	Backbone / Inspector	ChartMimic		
		Exec R.	Low L.	High L.
(a) Visual Encoder Ablation (using R_{vis} only)				
R_{vis}	SigLIP-so400m	95.5	79.2	88.7
	DINOv2-G	96.0	80.2	89.2
	WebSSL-1B	96.4	80.5	89.8
(b) Reward Mechanism Ablation (R_{vis} with WebSSL-1B)				
R_{judge}	Q3-VL-30B	95.5	79.9	89.0
R_{inq}		96.5	80.4	89.3
$R_{vis} + R_{judge}$		96.0	80.3	90.9
$R_{vis} + R_{inq}$ **(Ours)**		**96.7**	**81.7**	**91.6**

Qwen2.5-VL base model (Row 4) improves code execution but **fails to achieve holistic alignment**, lagging significantly behind the SFT baseline in visual fidelity (76.7 vs 86.4 High-Level). This confirms that **a robust SFT foundation** is essential for initializing the policy space to a learnable region. However, once properly initialized via SFT, our Inquiry-Driven RL substantially boosts performance, pushing the High-Level score to 91.6. Furthermore, this strategy proves effective even when applied to external baselines. Specifically, applying our RL stage to the **ChartMaster-7B** (Tan et al., 2025) checkpoint (Row 2) yields further performance gains. This demonstrates that our Inquiry-Driven reward is a versatile alignment framework capable of refining diverse foundations, regardless of their underlying training recipes.

4.3.2 Reward Ablation

We conduct an in-depth ablation analysis of the core reward mechanisms to validate our design

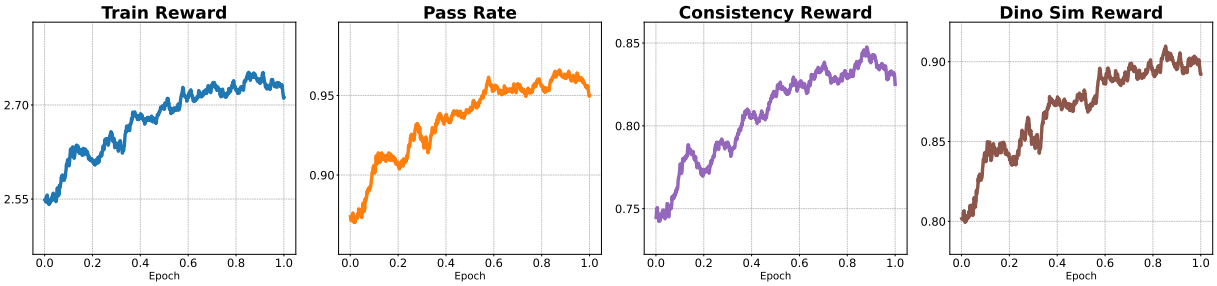

Figure 3: **Training dynamics of the Inquiry-Driven RL phase.** The Train Reward, Pass Rate, and Consistency Reward exhibit synchronized upward trajectories, indicating stable and effective optimization.

choices. The results are summarized in Table 4.

Visual Encoder Selection Section (a) evaluates the impact of different visual backbones when using a pure visual similarity reward (R_{vis}). We compare the semantic-focused **SigLIP-so400m** (Zhai et al., 2023) against two structure-aware SSL models: **DINOv2-G** (Oquab et al., 2023) and **WebSSL-1B** (Fan et al., 2025). We observe that WebSSL-1B achieves the highest Execution Rate (**96.4**) and Low-Level score (**80.5**). Although it shares the same DINO-based architecture as DINOv2, WebSSL derives significant benefits from being pre-trained on large-scale web data. This domain alignment makes it more effective than natural-image-based models in capturing the fine-grained structural nuances (e.g., grid lines, marker shapes) required for precise chart reproduction. We provide more qualitative comparisons in Appendix D.1.

Reward Strategies Comparison Section (b) validates the effectiveness of our Inquiry-Driven strategy (R_{inq}) compared to the VLM Judge baseline (R_{judge}). The comparison yields two key insights:

(1) ***Instability of VLM-as-a-Judge.*** Relying solely on the VLM judge (R_{judge}) yields the lowest performance across most metrics. Moreover, adding R_{judge} to R_{vis} causes regression in Low-Level metrics (80.5 $\rightarrow$ 80.3) and Execution Rate (96.4 $\rightarrow$ 96.0). This confirms that subjective, black-box VLM scoring introduces high variance and hallucination noise that hinder optimization.

(2) ***Superiority of Atomic Verification.*** Conversely, R_{inq} alone outperforms R_{judge}. Combined with visual constraints ($R_{vis}+R_{inq}$), the model achieves SOTA results (High-Level: **91.6**, Low-Level: **81.7**). By decomposing evaluation into atomic, objective VQA tasks, CharTide ensures fidelity and mitigates the instability of holistic VLM scoring.

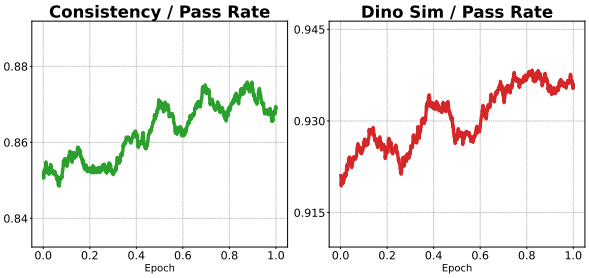

Figure 4: **Reward Hacking Analysis.** The continuous rise in rewards normalized by pass rate confirms that the model improves visual fidelity *per executable sample*, distinct from merely exploiting execution rates.

4.4 Analysis

4.4.1 Reward Hacking Examination

As shown in Figure 3, the Train Reward, Pass Rate, and Consistency Reward improve consistently throughout the training phase. To validate the efficacy of our RL stage and investigate potential reward hacking, where the model might maximize rewards by exploiting the execution rate at the expense of visual fidelity, we conduct an in-depth analysis of the training dynamics.

Specifically, we track the average reward per executable sample in Figure 4 to verify that the increase is not merely an artifact of generating more compilable code. As illustrated by the **Consistency / Pass Rate** and **Visual Reward / Pass Rate** curves, even when conditioning on executed samples, the metrics exhibit a steady ascent over epochs. This implies a dual improvement: the model generates more executable code (**Quantity** $\uparrow$) while simultaneously achieving *higher* visual fidelity within those valid samples (**Quality** $\uparrow$). This positive trend strongly refutes the reward hacking hypothesis, confirming that our Inquiry-Driven strategy successfully guides the model to optimize fine-grained visual grounding while ensuring syntactic correctness during the optimization process.

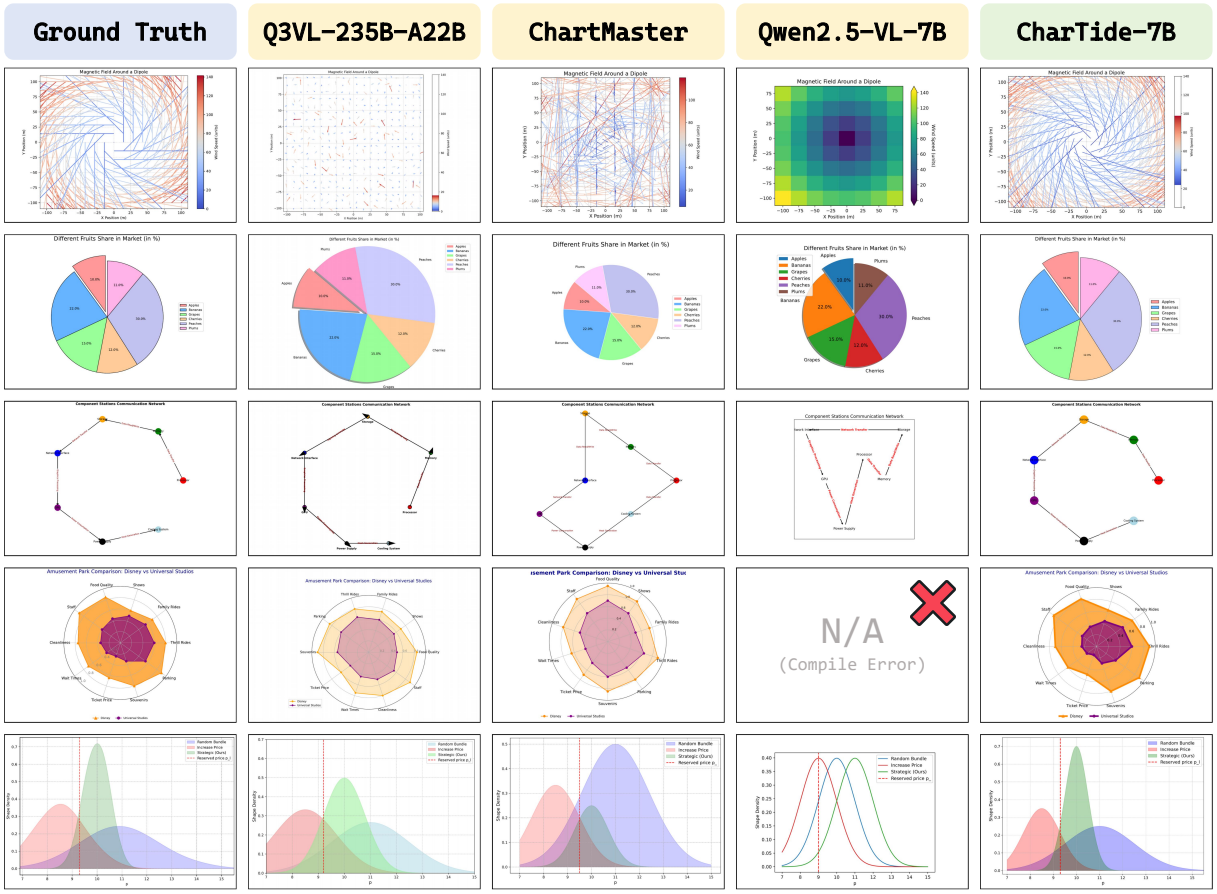

Figure 5: **Qualitative comparison on challenging chart types from ChartMimic Benchmark.** We compare CharTide against various baselines on topologically complex and information-dense samples. While baselines often suffer from structural collapse or omit fine-grained visual details, our model achieves high-fidelity reproduction aligned with the ground truth. We provide additional visualization examples in Appendix D.3.

4.4.2 Data Contamination Analysis

Although our Chart $\rightarrow$ Code training data is synthetically regenerated via Qwen3-VL-235B-A22B, theoretically preventing direct data leakage, we conduct a contamination analysis to rule out potential indirect leakage from the source ChartCap dataset. To ensure a robust assessment and mitigate the potential bias of a single feature extractor, we compute the cosine similarity based on the averaged feature embeddings from both WebSSL-1B and DINOv2-Giant encoders. We perform this retrieval between every sample in the ChartMimic test set ($N = 600$) and the entire 500k ChartCap dataset. Detailed visualizations of the retrieval results are provided in Appendix D.2.

Our analysis reveals that even for the pairs with the highest averaged similarity scores, the retrieved training samples merely share similar chart types or color schemes but contain distinct data values and semantic contexts. This confirms that CharTide achieves performance gains through robust style learning and our tri-perspective training strategy, rather than by memorizing benchmarks.

4.4.3 Qualitative Visualization

Figure 5 presents a qualitative comparison of complex charts, highlighting two key advantages:

Structural Integrity. For topologically complex charts such as quiver plots and graph structures, baseline models frequently exhibit structural hallucinations or incomplete generation. In contrast, CharTide faithfully reproduces vector field directions and intricate node connections, preserving the underlying topological logic.

Detail Perception. In dense scenarios like radar and density plots, baselines often generate rough outlines but struggle with fine-grained attributes like transparency and overlaps. Conversely, CharTide maintains high fidelity in color mapping, data trends, and text annotations, corroborating the quantitative gains reported in Table 1.

5 Conclusion

We propose CharTide, a unified data-centric framework for high-precision chart-to-code generation. By introducing the Tri-Perspective Tuning strategy, we effectively overcome the homogeneity bottleneck inherent in existing SFT data by explicitly decoupling visual perception from logical reasoning. Furthermore, our Inquiry-Driven RL mechanism transforms the alignment paradigm, shifting from subjective VLM scoring to objective, atomic QA verification. This ensures rigorous consistency between visual semantics and the generated code. Extensive experiments demonstrate that CharTide not only significantly outperforms SOTA open-source baselines but also achieves performance parity with top-tier proprietary models across authoritative benchmarks. We envision this work as a robust foundation for future advancements in more reliable multimodal code agents.

Limitations

Despite the comprehensive performance achieved by CharTide in resolving core challenges of chart-to-code generation, several avenues remain for future exploration. To ensure fair comparisons with mainstream open-source baselines, our experiments were primarily conducted within the 7B/8B parameter regime. While CharTide establishes a strong SOTA at this scale, the performance upper bounds accessible via larger backbones remain to be explored. Furthermore, our current evaluation concentrates on the *Direct Mimic* task. Although the model demonstrates robust logic synthesis, we have not extensively evaluated scenarios involving iterative chart modification, style transfer, or refactoring based on customized user data. Future work will aim to extend CharTide to broader applications, such as interactive editing and cross-library type migration, bridging the gap between static reproduction and dynamic creation.

Ethical Considerations

This work uses only publicly available datasets and open-source foundation models, following their specific licenses. Since the data is mostly scientific and synthetic charts, it contains no private personal details. Also, because the main goal is generating structured plotting code rather than general text, the risk of producing toxic or biased content is minimal. We believe this research helps make data visualization tools accessible to a wider audience, with no expected negative impact on society.

Acknowledgment

This work is supported by the Frontier Technologies R&D Program of Jiangsu (BF2024059) and the National Natural Science Foundation of China (Grant #62572229 and #62502544).

References

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025a. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025b. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Lei Chen, Xuanle Zhao, Zhixiong Zeng, Jing Huang, Liming Zheng, Yufeng Zhong, and Lin Ma. 2025a. Breaking the sft plateau: Multimodal structured reinforcement learning for chart-to-code generation. *arXiv preprint arXiv:2508.13587*.

Lei Chen, Xuanle Zhao, Zhixiong Zeng, Jing Huang, Yufeng Zhong, and Lin Ma. 2025b. Chart-r1: Chain-of-thought supervision and reinforcement for advanced chart reasoner. *Preprint*, arXiv:2507.15509.

Yang Chen, Yufan Shen, Wenxuan Huang, Sheng Zhou, Qunshu Lin, Xinyu Cai, Zhi Yu, Jiajun Bu, Botian Shi, and Yu Qiao. 2025c. Learning only with images: Visual reinforcement learning with reasoning, rendering, and visual feedback. *Preprint*, arXiv:2507.20766.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, and 1 others. 2024. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In *Proceedings of the 32nd ACM international conference on multimedia*, pages 11198–11201.

David Fan, Shengbang Tong, Jiachen Zhu, Koustuv Sinha, Zhuang Liu, Xinlei Chen, Michael Rabbat, Nicolas Ballas, Yann LeCun, Amir Bar, and 1 others. 2025. Scaling language-free visual representation learning. *arXiv preprint arXiv:2504.01017*.

Yi Gui, Zhen Li, Yao Wan, Yemin Shi, Hongyu Zhang, Bohua Chen, Yi Su, Dongping Chen, Siyuan Wu, Xing Zhou, and 1 others. 2025. Webcode2m: A real-world dataset for code generation from webpage designs. In *Proceedings of the ACM on Web Conference 2025*, pages 1834–1845.

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*.

Wei He, Zhiheng Xi, Wanxu Zhao, Xiaoran Fan, Yiwen Ding, Zifei Shan, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Distill visual chart reasoning ability from llms to mllms. *arXiv preprint arXiv:2410.18798*.

Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Yao Hu, and Shaohui Lin. 2025. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*.

Lingjie Jiang, Shaohan Huang, Xun Wu, Yixia Li, Dongdong Zhang, and Furu Wei. 2025a. Viscodex: Unified multimodal code generation via merging vision and coding models. *arXiv preprint arXiv:2508.09945*.

Yilei Jiang, Yaozhi Zheng, Yuxuan Wan, Jiaming Han, Qunzhong Wang, Michael R Lyu, and Xiangyu Yue. 2025b. Screencoder: Advancing visual-to-code generation for front-end automation via modular multimodal agents. *arXiv preprint arXiv:2507.22827*.

Jovana Kondic, Pengyuan Li, Dhiraj Joshi, Zexue He, Shafiq Abedin, Jennifer Sun, Ben Wiesel, Eli Schwartz, Ahmed Nassar, Bo Wu, and 1 others. 2025. Chartgen: Scaling chart understanding via code-guided synthetic chart generation. *arXiv preprint arXiv:2507.19492*.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, and 1 others. 2024. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.

Junyoung Lim, Jaewoo Ahn, and Gunhee Kim. 2025. Chartcap: Mitigating hallucination of dense chart captioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13171–13182.

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.

Ahmed Masry, Mohammed Saidul Islam, Mahir Ahmed, Aayush Bajaj, Firoz Kabir, Aaryaman Kartha, Md Tahmid Rahman Laskar, Mizanur Rahman, Shadikur Rahman, Mehrad Shahmohammadi, and 1 others. 2025. Chartqapro: A more diverse and challenging benchmark for chart question answering. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19123–19151.

Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Tiancheng Han, Botian Shi, Wenhai Wang, Junjun He, and 1 others. 2025. Mm-eureka: Exploring the frontiers of multimodal reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2503.07365*.

Minheng Ni, Zhengyuan Yang, Linjie Li, Chung-Ching Lin, Kevin Lin, Wangmeng Zuo, and Lijuan Wang. 2025. Point-rft: Improving multimodal reasoning with visually grounded reinforcement finetuning. *arXiv preprint arXiv:2505.19702*.

Tianhao Niu, Yiming Cui, Baoxin Wang, Xiao Xu, Xin Yao, Qingfu Zhu, Dayong Wu, Shijin Wang, and Wanxiang Che. 2025. Chart2code53: A large-scale diverse and complex dataset for enhancing chart-to-code generation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 15839–15855.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, and 1 others. 2023. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*.

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*.

Chenglei Si, Yanzhe Zhang, Ryan Li, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. 2025. Design2code: Benchmarking multimodal code generation for automated front-end engineering. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3956–3974.

Wentao Tan, Qiong Cao, Chao Xue, Yibing Zhan, Changxing Ding, and Xiaodong He. 2025. Chartmaster: Advancing chart-to-code generation with real-world charts and chart similarity reinforcement learning. *arXiv preprint arXiv:2508.17608*.

Yuxuan Wan, Chaozheng Wang, Yi Dong, Wenxuan Wang, Shuqing Li, Yintong Huo, and Michael Lyu. 2025. Divide-and-conquer: Generating ui code from screenshots. *Proceedings of the ACM on Software Engineering*, 2(FSE):2099–2122.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin

Jing, Shenglong Ye, Jie Shao, and 1 others. 2025. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*.

Chengyue Wu, Zhixuan Liang, Yixiao Ge, Qiushan Guo, Zeyu Lu, Jiahao Wang, Ying Shan, and Ping Luo. 2025. Plot2code: A comprehensive benchmark for evaluating multi-modal large language models in code generation from scientific plots. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 3006–3028.

Renqiu Xia, Hancheng Ye, Xiangchao Yan, Qi Liu, Hongbin Zhou, Zijun Chen, Botian Shi, Junchi Yan, and Bo Zhang. 2025. Chartx & chartvlm: A versatile benchmark and foundation model for complicated chart reasoning. *IEEE Transactions on Image Processing*.

Long Xing, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Jianze Liang, Qidong Huang, Jiaqi Wang, Feng Wu, and Dahua Lin. 2025. Caprl: Stimulating dense image caption capabilities via reinforcement learning. *arXiv preprint arXiv:2509.22647*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Cheng Yang, Chufan Shi, Yaxin Liu, Bo Shui, Junjie Wang, Mohan Jing, Linran Xu, Xinyu Zhu, Siheng Li, Yuxiang Zhang, and 1 others. 2024. Chartmimic: Evaluating lmm's cross-modal reasoning capability via chart-to-code generation. *arXiv preprint arXiv:2406.09961*.

Yi Yang, Xiaoxuan He, Hongkun Pan, Xiyan Jiang, Yan Deng, Xingtao Yang, Haoyu Lu, Dacheng Yin, Fengyun Rao, Minfeng Zhu, Bo Zhang, and Wei Chen. 2025b. R1-onevision: Advancing generalized multimodal reasoning through cross-modal formalization. *arXiv preprint arXiv:2503.10615*.

Sukmin Yun, Rusiru Thushara, Mohammad Bhat, Yongxin Wang, Mingkai Deng, Jinhong Wang, Tianhua Tao, Junbo Li, Haonan Li, Preslav Nakov, and 1 others. 2024. Web2code: A large-scale webpage-to-code dataset and evaluation framework for multimodal llms. *Advances in neural information processing systems*, 37:112134–112157.

Yuheng Zha, Kun Zhou, Yujia Wu, Yushu Wang, Jie Feng, Zhi Xu, Shibo Hao, Zhengzhong Liu, Eric P Xing, and Zhiting Hu. 2025. Vision-g1: Towards general vision language reasoning with multi-domain data curation. *arXiv preprint arXiv:2508.12680*.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. 2023. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11975–11986.

Kaiyan Zhang, Yuxin Zuo, Bingxiang He, Youbang Sun, Runze Liu, Che Jiang, Yuchen Fan, Kai Tian, Guoli Jia, Pengfei Li, and 1 others. 2025a. A survey of reinforcement learning for large reasoning models. *arXiv preprint arXiv:2509.08827*.

Zhihan Zhang, Yixin Cao, and Lizi Liao. 2025b. Boosting chart-to-code generation in mllm via dual preference-guided refinement. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 11032–11041.

Xuanle Zhao, Deyang Jiang, Zhixiong Zeng, Lei Chen, Haibo Qiu, Jing Huang, Yufeng Zhong, Liming Zheng, Yilin Cao, and Lin Ma. 2025a. Vincicoder: Unifying multimodal code generation via coarse-to-fine visual reinforcement learning. *arXiv preprint arXiv:2511.00391*.

Xuanle Zhao, Xianzhen Luo, Qi Shi, Chi Chen, Shuo Wang, Zhiyuan Liu, and Maosong Sun. 2025b. Chartcoder: Advancing multimodal large language model for chart-to-code generation. *arXiv preprint arXiv:2501.06598*.

A Evaluation Details

To ensure a strictly fair comparison, all results reported in Table 1 were re-evaluated using the official open-source weights under identical experimental settings. We did not rely on reported numbers from original papers, which may vary due to differences in evaluation environments or API versions. The specific protocols for each benchmark are as follows:

ChartMimic. We utilize the test set from the **ChartMimic v2** release (Yang et al., 2024), specifically the "Direct Mimic" task, which consists of 600 chart images. We conduct evaluations using the `vlmevalkit` (Duan et al., 2024) framework, adhering strictly to its default settings. The evaluation metrics consist of: (1) **High-Level Score (0–100):** Assessed by `GPT-4o-2024-05-13` to measure overall visual and semantic consistency; (2) **Low-Level Score:** Calculated as the average F1 score across four atomic dimensions: text, layout, chart type, and color. Results are reported as the mean of three independent measurements per test.

Plot2Code. We employ the official evaluation pipeline (Wu et al., 2025) on the standard test set comprising 133 scientific plots. Metrics include Code Execution Rate, Text Match, and Visual Similarity Rating. *Note on Re-evaluation:* (1) **Judge Model Update:** Since the original judge model, GPT-4V, has been deprecated, we replaced it with **GPT-4o** for the Visual Similarity Rating. Consequently, we re-ran evaluations for all baseline models using this upgraded judge to ensure a unified and fair comparison. (2) **Metric Normalization Adjustment:** The original Plot2Code benchmark computes Text Match and Rating scores only on *executable* samples, which artificially inflates the visual scores for models with low execution rates (survivorship bias). To address this, we modified the evaluation to calculate the average score over the **total number of test samples** (assigning zero to failed executions). This rigorous standard ensures a holistic assessment but may result in numerical discrepancies compared to prior literature.

ChartX. We focus on the **Chart Generation/Reproduction** task within the ChartX benchmark (Xia et al., 2025), which contains 1,152 chart images. Performance is measured using a GPT-4 based scoring metric (on a scale of 0–5), which evaluates the semantic alignment between the generated code logic and the visual attributes of the ground truth.

A.1 OOD Generalization Beyond Direct Mimic

Customized Mimic. We additionally evaluate zero-shot transfer on the Customized Mimic split of ChartMimic using the same evaluation toolkit and default settings as the Direct Mimic protocol. This setting is more challenging because the model must follow less literal reproduction instructions without any task-specific fine-tuning.

Seaborn OOD. To test backend generalization, we build a small OOD set from 47 official Seaborn example scripts. Because some low-level metrics in ChartMimic assume a matplotlib execution stack, we report execution rate, average score over all examples, and average score over executable examples. We use GPT-4o-2024-11-20 as the judge model for the high-level evaluation.

Table 5: **Detailed OOD evaluation results.**

Setting	Model	Exec.	Avg(all)	Avg(exec)
Customized Mimic	Qwen2.5-VL-7B	79.0	56.42	64.61
	CharTide-7B	**92.7**	**62.91**	**79.09**
Seaborn OOD	Qwen2.5-VL-7B	61.70	41.04	66.52
	CharTide-7B	**82.98**	**69.40**	**83.64**

A.2 Significance Test for Inquiry-Driven Reward

We perform a paired t-test over the 600 ChartMimic test samples, comparing $R_{vis} + R_{judge}$ against $R_{vis} + R_{inq}$.

Table 6: **Paired significance test for the proposed reward.**

Metric	Δ Mean	p-value	Interpretation
Overall	+1.41	0.027	Significant
Chart Type	+2.53	0.019	Significant
Layout	+1.58	0.055	Marginal
Color	+1.77	0.086	Marginal

In addition, 12.8% of the test samples (77/600) improve by more than 10 points, indicating that the gain is not merely a uniform minor shift.

B SFT Data Source and Pipeline Details

The core foundation of our training data is derived from **ChartCap** (Lim et al., 2025), a large-scale dataset containing 500k high-quality charts paired with detailed captions and info. We process this

data into three distinct streams to support our Tri-Perspective SFT strategy.

B.1 Visual Perception Stream (Chart → Caption)

For the visual perception stream, we directly leverage the image-caption pairs from ChartCap. To ensure compatibility with the context window of our base models and to remove potential noise from excessively long text, we apply a length-based filter, excluding samples where the caption length exceeds 4,096 tokens. This results in a high-quality dataset focused on dense visual description.

B.2 Code Logic Stream (Caption → Code)

To bolster the model's ability to synthesize code from pure textual logic, we construct a *Caption-to-Code* dataset.

Synthesis. We concatenate the descriptive caption and the structural 'info' (data table summary) from ChartCap to form a rich textual context. We then employ **Qwen3-Coder-30B-A3B** to generate the corresponding Python plotting code. The prompt used for this generation is as follows:

> **System Prompt:** You are an expert Python data visualization engineer.
> **User Instruction:** Based on the following chart description and data summary, please write Python code using Matplotlib/Seaborn to reproduce the chart.
> **Input Data:**
> `[Caption]: {input_caption}`
> `[Data Info]: {input_info}`
> **Constraints:** Ensure the code is self-contained, executable, and strictly follows the data trends described.

Filtering Pipeline. To ensure data quality, we execute the generated code to render a new chart image (I_{gen}). We then employ a rigorous consistency check using **Qwen3-VL-235B-A22B**. Specifically, we feed the original ground-truth image (I_{src}), the generated image (I_{gen}), and the original caption into the VLM, instructing it to evaluate whether I_{gen} semantically and visually aligns with I_{src} given the text description. After filtering out execution failures and low-consistency samples, we retain approximately **400k** high-quality pairs.

B.3 Modality Fusion Stream (Chart → Code)

This stream targets the end-to-end capability of converting visual charts directly into code.

Data Aggregation. We expand the data scale by combining the 500k images from ChartCap with an additional 500k charts collected from open-source datasets used in prior works (Zhao et al., 2025b,a)(with Apache-2.0 license). This yields a total pool of **1 million** chart images.

Code Generation. We utilize the powerful **Qwen3-VL-235B-A22B** model to generate ground-truth quality code for these images. The prompt is designed to enforce strict visual alignment:

> **System Prompt:** You are an expert in chart reverse engineering.
> **User Instruction:** Write Python code to reproduce this chart image using matplotlib.
> **Input Image:** [Image Placeholder]
> **Requirements:**
>
> 1. The code must be executable.
> 2. Use the exact data values visible in the chart.
> 3. Strictly mimic the styling (colors, fonts, legend position, markers).

Post-generation, we execute the code to render the predicted charts (I_{pred}). To eliminate visual hallucinations and ensure high reproduction fidelity, we implement a strict visual verification mechanism. We utilize the **WebSSL-1B**(Fan et al., 2025) encoder to extract visual feature embeddings from both the original ground-truth chart (I_{src}) and the rendered chart (I_{pred}). We then compute the cosine similarity between these embeddings and retain only those samples where the similarity score exceeds a threshold of **0.8**. This rigorous filtering process ensures that the final training data maintains high visual alignment standards.

B.4 Information Asymmetry Analysis

To quantify the supervision imbalance in end-to-end Chart→Code training, we randomly sample 5,000 Chart→Code examples (approximately 2.33M tokens in total) and categorize tokens into: (1) boilerplate template, (2) data definition, (3) visual configuration, (4) plotting calls, and (5) other code. We then further isolate the subset of visual attribute values that truly need to be inferred from the image, such as color, marker, fontsize, linewidth,

linestyle, and alpha.

Table 7: **Token composition of Chart→Code supervision.**

Category	Share	Description
Non-visual tokens	70.8%	imports, figure init, comments, data processing
Visual-related tokens	29.2%	titles, legends, ticks, plotting APIs
Visual attribute values	2.9%	color, marker, fontsize, linewidth, linestyle, alpha

The ratio between non-visual tokens and truly image-conditioned visual-attribute values is therefore approximately 25:1. Moreover, for several visual attributes (e.g., marker, fontsize, linewidth, and linestyle), the Top-3 values cover 63%–86% of occurrences. This confirms that standard cross-entropy training is naturally dominated by predictable scaffolding and common-value patterns, which encourages template memorization over fine-grained visual grounding.

C QA Data Generation Details

To construct a high-quality, verifiable reward model for the RL stage, we curated a specialized VQA dataset. The pipeline consists of three phases: representative sampling, question generation, and consistency filtering.

C.1 Representative Sampling

Given the redundancy in the large-scale pre-training data, we aim to select a diverse subset for reinforcement learning. We first utilize the **WebSSL-1B** encoder to extract high-dimensional visual feature vectors from the 500k charts in the ChartCap dataset. Subsequently, we apply K-Means clustering on these features to identify distinct visual clusters. From these clusters, we sample **30k** representative chart images, ensuring coverage of diverse chart types, layouts, and data distributions.

C.2 Question Generation

For each selected chart, we leverage **GPT-5** to generate a set of 10 evaluation questions ($N = 10$). To ensure the ground truth answers are accurate, we provide GPT-5 with both the rich semantic captions and the underlying data tables (info) from ChartCap as context. The system prompt used to guide GPT-5 is designed to cover multiple dimensions of chart assessment:

System Prompt: You are a Data Visualization Benchmark Creator. Your task is to generate **10 Evaluation Questions** to judge if a reconstructed chart matches the original.

DATA SOURCE RULES:

1. **DATA & TEXT:** TRUST THE CAPTION. If caption says "16%", answer is 16.
2. **STYLE & LAYOUT:** LOOK AT THE IMAGE for visual details missing in text.

DIFFICULTY SETTINGS:

- **Keep it Simple:** Focus on obvious data retrieval (e.g., "What is the max value?") rather than complex math.
- **Generous Tolerance:** For numerical questions, allow a wider margin of error (e.g., if value is 50, tolerance should be 5.0). The goal is to check if the trend is correct, not pixel-perfect precision.

QUESTION DISTRIBUTION (Total 10):

- **Chart Type (1 Q - Bool):** Is it the correct chart type? (e.g., "Is this a stacked bar chart?")
- **Layout (1 Q - Bool):** Check subplot arrangement or legend position.
- **Text Content (3 Qs - Bool):**
  - 2 Positive checks (e.g., "Is X-axis label 'Year'?").
  - 1 Negative/Trick check (e.g., "Does title contain '2050'?" → No).
- **Data Accuracy (3 Qs - Float):**
  - Ask for 3 distinct data values mentioned in the caption.
  - **CRITICAL:** Set a 'tolerance' of at least 5-10% of the value.
- **Style (2 Qs - Bool):** Check Colors or Marker shapes (e.g., "Are bars red?", "Is line dashed?").

User Input: Ground Truth Caption: {caption}
Generate the 10 QA pairs now.

C.3 Consistency Filtering

To mitigate potential hallucinations in the generation process and ensure the questions are answerable by a visual inspector, we implement a validation step. We employ **Qwen3-VL-30B-A3B** to answer the generated questions based on the original chart images. We verify the model's answers against the GPT-5 generated ground truth, applying the specified numerical tolerance for data-related questions. We filter out any charts where the model fails to answer correctly, retaining approximately **20k** high-quality, verified QA samples for the final RL training set.

D More Qualitative Analysis

D.1 Qualitative Analysis of Visual Models

As illustrated in Figure 6, to intuitively validate the superiority of WebSSL-1B in assessing chart-to-code reconstruction quality, we qualitatively analyzed representative pairs from the Modality Fusion filtering pipeline.

We observed that standard visual encoders often fail to penalize subtle structural errors or semantic deviations if the overall color histogram remains similar. In contrast, WebSSL-1B demonstrates remarkable alignment with human perception by strictly penalizing structural collapses and fine-grained visual discrepancies (e.g., marker shapes, line styles), while implicitly capturing data trend deviations to effectively filter out "plausible but wrong" samples. These characteristics confirm that WebSSL-1B serves as a robust discriminative filter for our data construction and RL training.

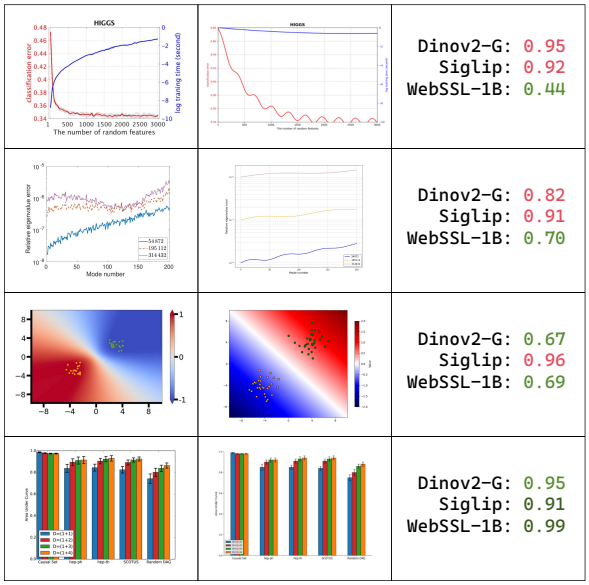

Figure 6: **Qualitative comparison of visual similarity matching.** We display sample pairs from the SFT filtering stage. The scores assigned by WebSSL-1B (shown above each pair) correlate strongly with the visual and structural consistency observed by human judges, effectively identifying high-quality reproductions while rejecting structural hallucinations.

D.2 Data Contamination Analysis

To ensure that our model's high performance stems from learning generalized plotting rules rather than simply memorizing the training data, we conducted a thorough data contamination check.

We computed the visual similarity between every image in the ChartMimic test set ($N = 600$) and the ChartCap 500k training dataset. To ensure a robust assessment, we employed two distinct encoders: **WebSSL-1B** and **DINOv2-Giant**. Specifically, we calculated the cosine similarity scores independently for each encoder and then averaged these scores to determine the final ranking. Based on these averaged scores, we retrieved the most similar training samples for each test case to identify potential leakage.

Figure 7 presents the "worst-case" scenario, showing the 6 test images that obtained the highest averaged similarity scores with the training set. We observe that while these pairs share a high degree of stylistic similarity, such as using standard Matplotlib color palettes, default layouts, or identical chart types, they are not duplicates. Notably, the actual content, including data values, axis ranges, and textual labels, remains different. This indicates that while the model has seen similar styles, it has not seen the specific data used in the test set.

For a broader perspective, Figure 8 shows 6 randomly selected test images and their nearest training neighbors based on the averaged metric. In this average case, the retrieved training images are visually and semantically distinct from the test queries, confirming that the vast majority of the test set has no close counterparts in the training data. These results confirm that CharTide achieves its performance through style generalization and robust instruction following, rather than data leakage.

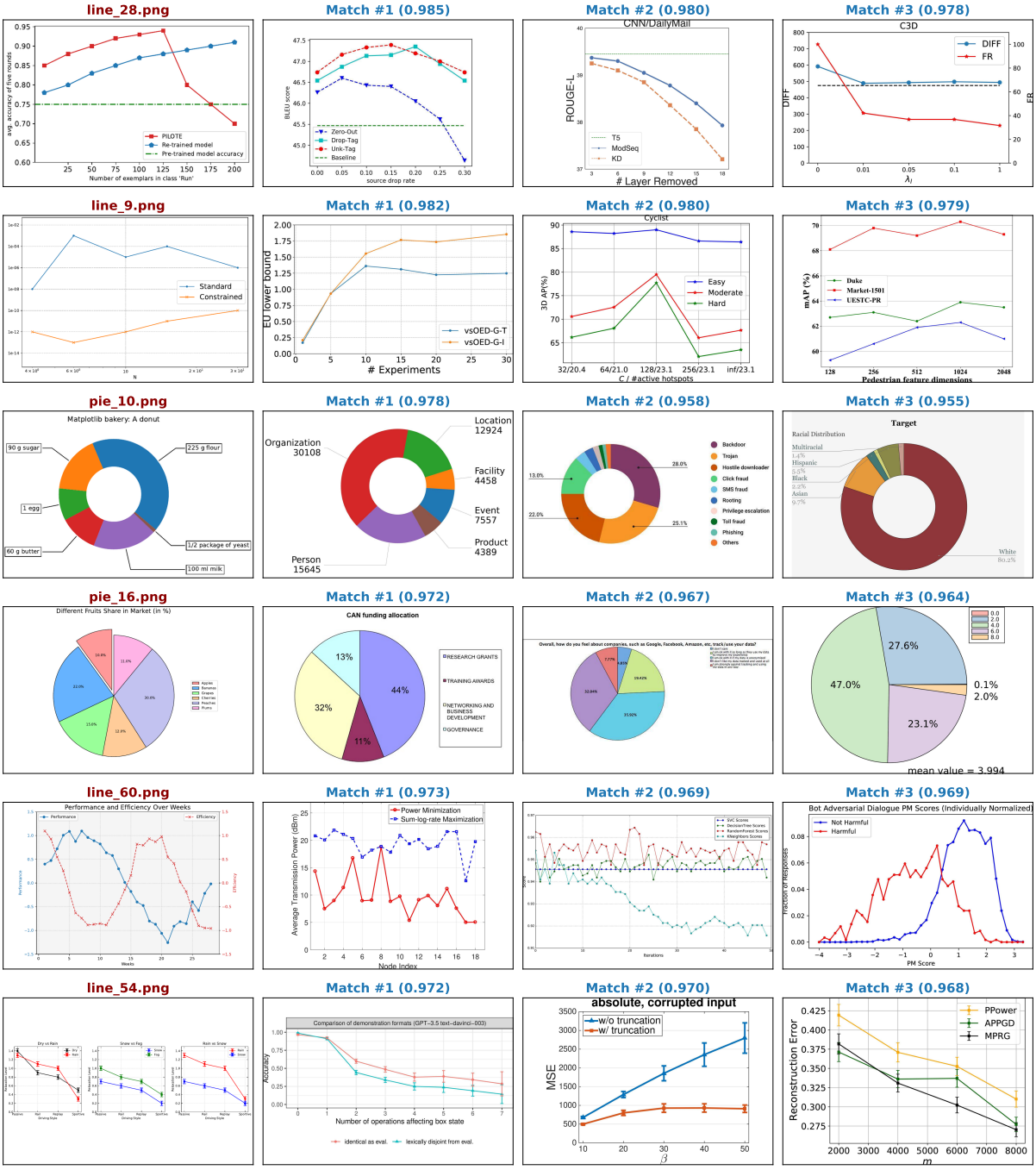

Figure 7: **Worst-case leakage check.** We display the 6 ChartMimic test samples with the *highest* similarity scores against the chartcap training set. The retrieved training samples (right) visually resemble the queries (left) in style and layout but differ in specific content and data values, confirming no direct leakage exists.

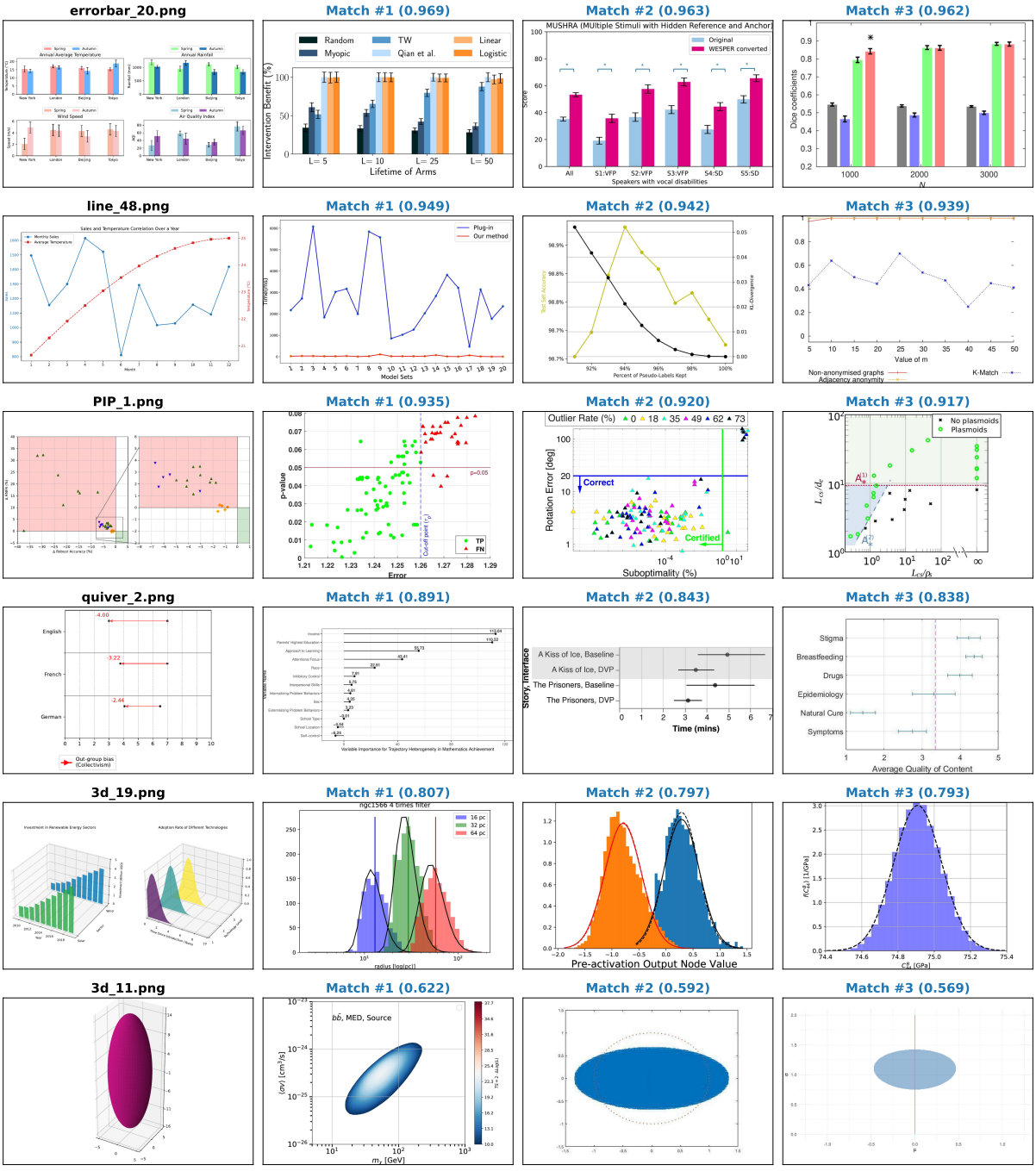

Figure 8: **Average-case leakage check.** We visualize 6 uniformly sampled ChartMimic images and their nearest neighbors in the chartcap training set, showing significant semantic and visual differences.

D.3 More Visualizations

Figure 9: More visualization results on ChartMimic benchmark. **"N/A"** in the figure indicates that the code generated by the model failed to compile, resulting in no output image. [1 / 3].

Figure 10: More visualization results on ChartMimic benchmark. **"N/A"** in the figure indicates that the code generated by the model failed to compile, resulting in no output image. [2 / 3].

Figure 11: More visualization results on ChartMimic benchmark. **"N/A"** in the figure indicates that the code generated by the model failed to compile, resulting in no output image. [3 / 3].