CycleChart: A Unified Consistency-Based Learning Framework for Bidirectional Chart Understanding and Generation

Dazhen Deng, Sen Yang, Yuchen He, Yuan Tian, and Yingcai Wu

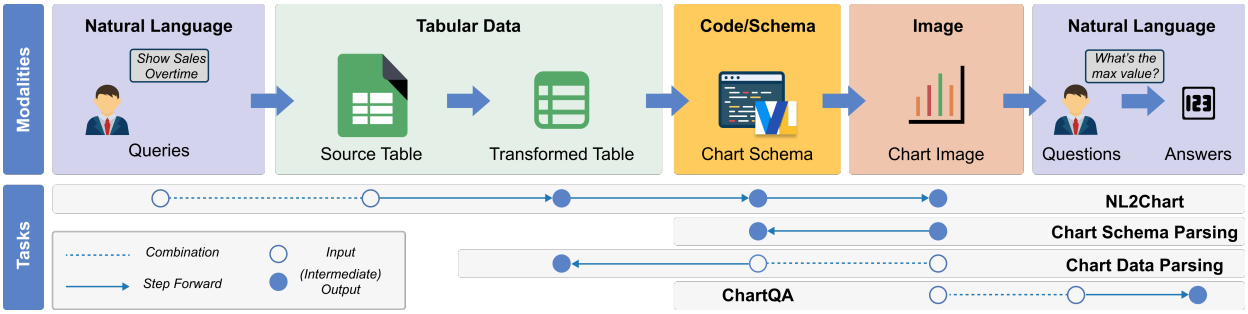

Fig. 1: The chart lifecycle spans four modalities (natural language, tabular data, code/schema, and image), forming a bidirectional pipeline that connects chart generation and chart understanding. NL2Chart traverses the forward path from a query and source table to a rendered chart; Chart Schema Parsing and Chart Data Parsing invert this path, recovering the specification and data from the image; ChartQA reasons over the chart to produce answers. CycleChart unifies all four tasks within a single model and enforces generate–parse consistency across the forward and reverse directions, bridging modalities into a closed cycle.

Abstract—Current chart-related tasks, such as chart generation (NL2Chart), chart schema parsing, chart data parsing, and chart question answering (ChartQA), are typically studied in isolation, preventing models from learning the shared semantics that link chart creation and interpretation. We introduce CycleChart, a consistency-based learning framework for bidirectional chart understanding and generation. Unlike conventional multi-task approaches that draw training samples independently across tasks, CycleChart organizes all tasks around each single data instance. From a source table and natural-language query, the model generates a chart specification, renders and executes it, then learns to recover the schema and underlying data from the resulting chart image. This per-instance lifecycle design lets the model capture the full chain of transformations, from raw data through visual encoding to structured recovery, and a generate–parse consistency objective enforces semantic alignment between the forward generation and reverse parsing directions. To support this framework, we construct CycleChart-Bench, a lifecycle-aligned benchmark where every chart sample carries aligned annotations for generation, schema parsing, data parsing, and question answering. CycleChart achieves strong results across all four tasks and transfers effectively to unseen external benchmarks, demonstrating improved cross-task generalization and marking a step toward more general chart understanding models.

Index Terms—Chart understanding, Chart generation, Consistency learning, Multimodal models.

1 Introduction

Charts encode numerical and categorical relationships through visual variables that together convey analytical semantics. Consequently, chart understanding and generation serve as the cornerstones of human-AI teaming for data analysis, as they determine whether models can not only interpret these visual artifacts and the embedded data insights, but also infer users' analytical intents. Early chart understanding models, constrained by limited vision capabilities, relied heavily on OCR or component detection to provide explicit structural cues for reasoning. Similarly, chart generation has long been formulated as predicting chart parameters or code, without modeling how data semantics are transformed into visual encodings or why certain visual designs communicate specific analytical intents.

While recent advances in chart-related tasks [12] have demonstrated remarkable progress, these approaches still lack a unified semantic understanding of charts, as they typically model each task independently. Chart Question Answering (ChartQA) [20, 21, 32, 35, 49] focuses on answering natural-language questions based on chart images, requiring both visual perception and numerical reasoning. Chart Parsing [13, 24] aims to recover structured representations—such as chart schemas or underlying data tables—from rendered visualizations. Chart Generation [18, 19, 26, 29, 31] involves producing visual specifications or rendered charts from data tables or natural-language inputs. To overcome such task fragmentation, the emergence of large multimodal models has motivated researchers to unify different chart-related tasks under a shared framework [6, 41]. However, these approaches mainly leverage the models' ability to handle multimodal inputs and outputs, rather than fully exploiting the intrinsic information in charts. This raises a fundamental question: how can models move beyond multimodal alignment to genuinely capture the internal logic of charts?

From the perspective of visualization theory, charts can be viewed through the lens of the Grammar of Graphics (GoG [37]), which defines a pipeline linking data and visual representations. In this grammar, a data table is first transformed into analytical variables, which are then mapped to visual marks and channels to produce the final chart. The chart, in turn, conveys information that humans interpret into facts or insights (Figure 1). Each corresponds to a different stage along the data-visual-semantic pipeline: chart generation encodes data into visual forms, chart parsing decodes visuals back into structured representations, and chart question answering interprets visuals to extract semantic insights. In practice, most datasets sample examples from one of these stages–either by generating charts from code or by collecting chart-question pairs from the web or academic publications–thus providing

• *Dazhen Deng, Sen Yang, Yuchen He, Yuan Tian, and Yingcai Wu are with Zhejiang University. E-mail: {dengdazhen, 22451267, heyuchen, yuantian, ycwu}@zju.edu.cn.*

Manuscript received xx xxx. 202x; accepted xx xxx. 202x. Date of Publication xx xxx. 202x; date of current version xx xxx. 202x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.202x.xxxxxxx

only partial supervision of the overall semantics. As a result, current models learn fragmented mappings within the data-visual-semantic continuum, rather than understanding the complete reasoning process that connects data, visual encodings, and analytical interpretation.

This insight motivates a unified and self-consistent framework that leverages the supervision signals inherent in the chart creation process. We propose **CycleChart**, a consistency-based learning framework that teaches MLLMs to perform both forward generation (from data and language to charts) and reverse parsing (from charts back to schema, data, and semantics) within a unified cycle. Given natural language and data tables, the model first generates a chart schema and executes it through a deterministic renderer, producing a chart image and its visualization-level table. These execution outputs then serve as automatically derived supervision targets: the model must recover the schema and data from the rendered chart, and the reconstruction loss enforces consistency between the generated and recovered representations. Because the reverse-path targets are produced by executing the chart specification through a deterministic renderer rather than by manual annotation, this cycle yields automatically derived supervision that scales with the training data without additional labeling effort.

To support this framework, we introduce **CycleChart-Bench**, a benchmark that covers the *full chart lifecycle*–from NL-driven chart generation to schema and data parsing, and finally to chart question answering. CycleChart-Bench further includes both single-view and faceted multi-view charts, making generation and reasoning tasks substantially more challenging by requiring precise localization and comparison of values across multiple subplots. Evaluations on a wide range of open- and closed-source models show that these faceted, lifecycle-aligned samples expose significantly harder cases than prior benchmarks, demonstrating CycleChart-Bench as a rigorous and challenging testbed for chart understanding. The contributions are as follows.

- **CycleChart:** A consistency-based learning framework that unifies chart generation, schema parsing, data parsing, and question answering through a closed generate-parse cycle, where reverse-path supervision is automatically derived from renderer execution without manual annotation.
- **CycleChart-Bench:** A lifecycle-aligned benchmark that links generation, parsing, and question answering within a cyclic loop, supporting both single-view and faceted charts.
- **Insights:** Enforcing generate–parse consistency yields three findings: (i) per-instance alignment, not task diversity, drives the gains; (ii) the cycle objective benefits compositionally complex faceted charts most; and (iii) the framework transfers to unseen chart styles (CharXiv, nvBench VQL) without architecture changes.

2 Related Work

This section introduces the related studies on chart understanding, chart generation, and consistency-based learning.

2.1 Chart Understanding

Chart understanding bridges computer vision and natural language processing, with chart parsing as a major subtask. Chart parsing typically involves three components: component extraction, schema prediction, and data extraction. Component extraction detects axes, marks, labels, and text, usually via object detection and OCR. Schema prediction [2, 27] aims to recover the chart's structural specification, expressed either as textual descriptions [31] or formal grammars such as matplotlib, Vega-Lite, or ggplot2. Data extraction [24] is the most challenging step, requiring models to infer the precise values encoded by visual marks relative to chart axes.

Beyond structured parsing, another major research direction focuses on the textual interpretation of charts. Chart captioning [11, 30] generates natural-language summaries but suffers from ambiguity and loosely defined outputs. ChartQA addresses this limitation by answering questions about chart images through visual grounding and numerical reasoning [16, 20, 21, 23, 25, 32, 35, 36, 40, 42, 43, 45], covering tasks from simple chart-type identification to multi-step numerical reasoning over visual elements.

Recent unified frameworks aim to handle multiple chart-understanding tasks within a single model [22, 48]. UniChart [22] introduces chart-specific pretraining tasks for both low-level element extraction and high-level reasoning, while ChartMoE [44] extends chart understanding to editing, replotting, highlighting, and transformation via natural-language instructions. However, these methods largely focus on interpreting existing charts and do not complete the full semantic cycle connecting data, chart generation, and chart understanding. Consequently, the bidirectional link between data, visualization structure, and natural-language semantics remains underexplored.

2.2 Chart Generation

Chart generation [46] aims to predict chart schemas from data tables, either with or without natural-language descriptions. When no textual intent is provided, the task becomes a chart recommendation, where the model analyzes a table to identify meaningful variable combinations and propose suitable visualizations [5, 38, 39]. With natural language guidance, recent work has increasingly focused on NL-driven chart generation [9], progressing from syntactic parsing approaches [26] to instruction-tuned LLMs trained on curated datasets and reward models [18, 19, 29, 31], enabling models to better translate analytic intents into chart specifications.

In addition, recent multimodal efforts have started to bridge chart understanding and chart generation within a single framework. ChartLlama [6] aligns chart, text, and data representations so that a multimodal LLM can perform comprehension tasks (e.g., QA, captioning, summarization) alongside generative tasks such as chart synthesis and editing. Similarly, multi-task benchmarks like ChartVLM [41] evaluate perception, reasoning, and redrawing jointly, suggesting the value of integrated modeling. These works point toward integrated solutions but still leave open how to enforce stronger bidirectional consistency between generation and parsing—an issue we address with CycleChart.

2.3 Consistency-based Learning

A key idea in consistency-based learning is to enforce that applying a forward and reverse mapping sequentially reconstructs the input, encouraging invertible, structure-preserving transformations. CycleGAN [50] introduced this idea for unpaired image translation, inspiring extensions such as DualGAN [47], ReCycleGAN [1], and dual learning in machine translation [7]. The same principle has been applied to cross-modal tasks, where cycle-consistency aligns vision and language representations [4, 33]. These methods operate in continuous spaces and require a differentiable round-trip loss through the pixel or feature domain. Charts, by contrast, are produced by executing a discrete specification through a deterministic renderer, making gradient-based cycle losses inapplicable and calling for a data-level consistency formulation.

In chart reasoning, grounding and consistency have been used to improve reliability. RefChartQA [32] introduces visual grounding for fine-grained chart QA, and Huang *et al.* [13] align chart captions with underlying data to correct factual errors. ChartPoint [42] further bridges visual perception and reasoning by integrating reflective interactions and re-rendered bounding boxes into chain-of-thought reasoning. However, these studies remain one-directional—verifying text or visuals separately—without enforcing bidirectional semantic consistency between chart generation and understanding. Our work explores this direction by applying generate–parse consistency at the data and specification level rather than in pixel space.

3 Problem Formulation

We consider four closely related tasks that together cover the full chart creation and understanding lifecycle: NL2Chart, chart schema parsing, chart data parsing, and ChartQA. We first introduce the shared notation and then define each task. We distinguish two types of tables throughout: the *raw table* t^{raw} is the original source data, while the *visualization-level table* t^{vis} is the result of applying data transformations (e.g., aggregation, filtering, binning) specified by the chart schema, which contains exactly the values encoded by the visual marks in the rendered chart.

Let I denote the space of chart images (e.g., rasterized plots), T the space of structured data tables, Q the space of natural-language queries or instructions, S the space of chart schemas (e.g., Vega-Lite specifications), and A the space of answers. We assume a deterministic rendering pipeline that takes a schema and the raw table, internally applies the data transformations specified by the schema to produce a visualization-level table, and render the final chart image:

$$R : S \times T \to I \times T, \quad (\hat{\imath}, t^{\text{vis}}) = R(s, t^{\text{raw}}).$$

Here t^{vis} is a byproduct of the rendering execution, not a separate input. In our Vega-Lite instantiation, we extract t^{vis} by intercepting the transformed data from the Altair runtime.

Natural Language to Chart Generation (NL2Chart). Given a raw table t^{raw} and a natural-language query q describing the desired analytical intent, the model generates a chart schema $\hat{s}$:

$$f_{\text{NL2Chart}} : T \times Q \to S, \quad \hat{s} = f_{\text{NL2Chart}}(t^{\text{raw}}, q). \tag{1}$$

In the Vega-Lite instantiation, $\hat{s}$ is a declarative specification that includes built-in data transformations (e.g., aggregation, filtering). Executing $\hat{s}$ through the renderer yields the chart image and its visualization-level table: $(\hat{\imath}, \hat{t}^{\text{vis}}) = R(\hat{s}, t^{\text{raw}})$.

Chart Schema Parsing. Chart schema parsing aims to recover the schema from a rendered chart image. In practice, the model also receives the table schema (column names and types) as auxiliary context, simulating a realistic scenario in which a user possesses the source dataset and wants to recover *how* it was visualized:

$$f_{\text{schema}} : I \times T_{\text{meta}} \to S, \quad \tilde{s} = f_{\text{schema}}(\hat{\imath}, t_{\text{meta}}). \tag{2}$$

The goal is for the predicted schema $\tilde{s}$ to match the visual schema s^{vis} (the gold specification with non-visual transform operations removed) up to permissible structural variations.

Chart Data Parsing. Chart data parsing further extracts the visualization-level table from the chart. Since the table depends on the schema's encodings (e.g., mark type, axes, facets), the task takes both the image and schema as input:

$$f_{\text{data}} : I \times S \to T, \quad \tilde{t}^{\text{vis}} = f_{\text{data}}(\hat{\imath}, \hat{s}). \tag{3}$$

The supervision target is the visualization-level table t^{vis} extracted from the Vega-Lite rendering pipeline.

Chart Question Answering (ChartQA). Given a chart image $\hat{\imath}$ and a natural-language question q about the chart, the model predicts a textual answer:

$$f_{\text{QA}} : I \times Q \to A, \quad \hat{a} = f_{\text{QA}}(\hat{\imath}, q_{\text{QA}}). \tag{4}$$

ChartQA requires both visual grounding and numerical or relational reasoning over the content encoded in the chart.

From tasks to cycle. The four tasks above are not independent: the output of forward generation ($\hat{s}$) is executed by the renderer to produce the inputs for all three reverse tasks ($\hat{\imath}, t^{\text{vis}}$), and the reverse tasks in turn verify whether the generated chart faithfully encodes the intended structure and semantics. This bidirectional dependency forms a closed generate–parse cycle that we exploit as a consistency-based training objective in section 5.

4 CycleChart-Bench Construction

To support our cycle-consistent training framework, we build *CycleChart-Bench*, a multi-task benchmark that provides aligned supervision for NL2Chart, schema parsing, data parsing, and ChartQA.

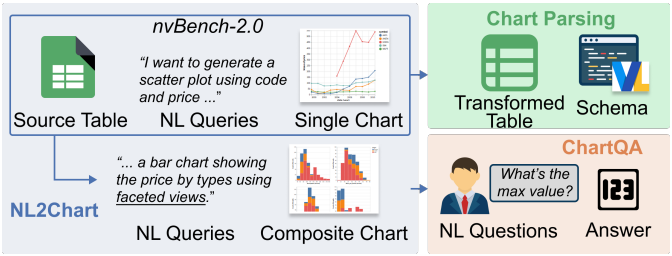

Fig. 2: CycleChart-Bench construction. nvBench 2.0 queries produce single-view charts, while our facet-oriented queries generate composite charts. Both chart types supply transformed tables and QA pairs for aligned parsing and reasoning supervision.

Motivation. Existing multi-task chart datasets often assemble tasks from heterogeneous sources, leading to weak coupling between a chart image, its underlying table, generation specification, and QA annotations. Such partial supervision prevents models from learning how chart creation, structure recovery, and reasoning relate to one another.

CycleChart-Bench instead models the *entire* lifecycle. For each sample, we begin with a raw table and an NL query that specifies an analytic intent, derive a chart specification, and execute it through the Vega-Lite renderer to produce the chart image and its visualization-level table. Parsing and QA labels are then derived automatically from these execution outputs: the specification serves as the schema parsing target, the visualization-level table as the data parsing target, and QA pairs are generated programmatically from the table values. This construction requires no manual annotation of parsing or QA labels, ensuring tight cross-task alignment while remaining fully scalable.

Beyond providing automatically derived targets, the benchmark serves two roles that an online cycle alone cannot fulfill. First, it supplies verified gold specifications that guarantee every training sample can complete the full generate–parse loop. Second, pre-computing the execution outputs offline avoids invoking the Vega-Lite renderer at every training iteration, making large-scale cycle-consistent training practical.

Based on nvBench 2.0. Our benchmark is constructed on nvBench 2.0 [18], which provides raw tables, NL queries, and Vega-Lite specifications for NL2Chart. nvBench's diverse chart types and intent-driven queries make it a strong base, but it lacks parsing labels and QA supervision. CycleChart-Bench extends each example into a full cycle:

$$(t^{\text{raw}}, q) \to (t^{\text{vis}}, s, \hat{\imath}) \to \text{parsing labels} \to \text{QA pairs}.$$

Facet-Augmented Charts. To reflect real-world multi-view layouts, we augment the nvBench charts by creating faceted versions when suitable categorical fields are present. These composed charts share data columns across subplots and enable cross-view reasoning, increasing structural diversity beyond single-view plots.

Visualization-Level Tables. For each (possibly faceted) Vega-Lite specification, we obtain the visualization-level table t^{vis} by executing the spec with Altair and intercepting the transformed data used for mark rendering. This yields precise supervision for chart data parsing.

ChartQA Generation. QA annotations are produced programmatically from t^{vis}, covering both single-view reasoning (extrema, comparisons, simple aggregations) and cross-facet reasoning for composed charts. LLMs are used only to generate natural-language templates; all answers are computed deterministically to avoid hallucination.

Statistics and validation. CycleChart-Bench contains 6,507 charts in total: 5,372 single-view and 1,135 faceted charts with five common mark types. We use an 8:1:1 train/validation/test split (5,205 / 650 / 652 samples), preserving the ratio of single- and multi-view charts. All Vega-Lite specifications are compiled via `vl-convert` to verify renderability; any specification that fails compilation is discarded. QA answers are computed deterministically from t^{vis}, avoiding hallucinated

labels. During evaluation, invalid predicted schemas receive a zero score for both validity and visual similarity metrics.

4.1 Construction Details

Chart type expansion and faceted chart augmentation. We first filter out unsupported or non-standard visualization types in nvBench 2.0, including boxplots and malformed specifications that fail to compile or do not conform to Vega-Lite conventions. We then programmatically augment eligible specifications into faceted charts. For each original Vega-Lite spec, we identify categorical fields that can partition the visualization and inject corresponding facet encodings (e.g., `"column"` or `"row"`) directly into the `encoding` block. All augmented specifications are rendered with `vl-convert` and validated through compile-time and runtime correctness checks. This procedure yields 1,135 high-quality faceted charts, which substantially increase structural diversity and introduce multi-view reasoning challenges.

Parsing labels. For schema parsing, we remove all `transform` operations from the original specification. These transformations (e.g., grouping, aggregation) modify the intermediate data but are not visually observable and therefore cannot be inferred from the image alone. The resulting schema s^{vis} represents the minimal visual structure needed for accurate parsing supervision. For data parsing, we execute the specification's transformation pipeline via `pandas` and serialize the result as CSV, capturing exactly the values encoded by the chart marks.

ChartQA annotations. We generate QA pairs using a large VLM (Qwen3-VL-235B). The model is provided with (i) the rendered image, (ii) the full Vega-Lite schema, (iii) the ground-truth data table produced by our parsing pipeline, and (iv) several seed questions. It is then instructed to produce new question–answer pairs that follow the same reasoning patterns while remaining faithful to the chart content. Seed questions are retrieved using a RAG pipeline: we embed 1,000 reasoning questions from CharXiv using `intfloat/multilingual-e5-large` and build a vector store. For each chart, we query the store using its NL intent to obtain the top-3 semantically relevant questions as stylistic exemplars. All final answers are verified deterministically against t^{vis}: we recompute each answer from the visualization-level table and discard any QA pair whose VLM-generated answer deviates from the recomputed value. This procedure retains approximately 78% of the generated pairs, ensuring that every retained answer is grounded in the chart data rather than hallucinated by the VLM.

5 CycleChart Training Framework

CycleChart trains a single multimodal model to jointly perform chart generation and chart understanding within a closed generate–parse cycle. Each training iteration operates on a single data instance and traverses both directions of this cycle, as illustrated in Figure 3: the *forward* path generates a chart from data and language, while the *reverse* path recovers the specification, data, and semantics from the rendered chart image. Crucially, the supervision targets for the reverse path are not manually annotated: they are automatically derived by executing the chart specification through a deterministic renderer, eliminating the need for additional labeling.

Forward: Chart Generation (NL2Chart). Given a raw table t^{raw} and a natural-language query q, the model predicts a chart schema $\hat{s}$, supervised by a token-level cross-entropy loss against the ground-truth specification:

$$L_{\text{NL2Chart}} = \text{CE}(\hat{s}, s).$$

We additionally validate whether $\hat{s}$ is syntactically correct and renderable; only valid schemas proceed to the reverse path, because an unrenderable specification cannot produce the chart image and visualization-level table on which the consistency cycle depends. For valid schemas, the Vega-Lite renderer R executes $\hat{s}$ on the raw table t^{raw}, producing a chart image and its visualization-level table as execution byproducts: $(\hat{\imath}, \hat{t}^{\text{vis}}) = R(\hat{s}, t^{\text{raw}})$. Both $\hat{\imath}$ and $\hat{t}^{\text{vis}}$ then serve as the input and supervision targets for the reverse path. In practice, CycleChart-Bench pre-computes these execution outputs offline, avoiding renderer invocations during training and enabling efficient cycle-consistent learning at scale (see section 4).

Reverse: Chart Parsing and Reasoning. Starting from the rendered chart image $\hat{\imath}$, the model performs three tasks that invert or interpret the forward generation. The reverse-path targets, which are the visual schema s^{vis} (the gold specification with non-visual `transform` operations removed; see section 4) and the visualization-level table t^{vis}, are both renderer-derived execution byproducts, as described in the opening paragraph.

Chart Schema Parsing. The model recovers a schema $\tilde{s}$ from $\hat{\imath}$. The target s^{vis} retains only the visually observable structure of the gold specification, so the reconstruction loss enforces generate–parse consistency at the level of visual encoding:

$$L_{\text{Schema}} = \text{CE}(\tilde{s},\, s^{\text{vis}}).$$

Chart Data Parsing. Conditioned on the rendered image and the visual schema, the model predicts a visualization-level table $\tilde{t}^{\text{vis}}$. The target t^{vis} is extracted from the Vega-Lite execution pipeline:

$$L_{\text{Data}} = \text{CE}(\tilde{t}^{\text{vis}},\, t^{\text{vis}}).$$

Together with L_{Schema}, this reconstruction loss completes the full generate–parse cycle, requiring the model to faithfully recover both the visual structure and the underlying data from the rendered chart. During training, the gold visual schema s^{vis} is provided as input to data parsing; at test time, the model uses its own predicted schema instead. This gap is mitigated in practice because schema parsing already achieves high accuracy (see Table 2), limiting error propagation.

ChartQA. Given the chart image and a QA query q_{QA}, the model predicts an answer $\hat{a}$, supervised by:

$$L_{\text{QA}} = \text{CE}(\hat{a}, a).$$

Schema and Data Parsing verify that the model can faithfully recover the *structural* content of a chart, i.e., its visual encoding and underlying values. ChartQA complements them by testing *semantic* consistency: whether the model can derive correct analytical conclusions from the same visual artifact. Together, the three reverse tasks form a progression from structure to semantics, ensuring that the generate–parse cycle enforces consistency at every level of chart interpretation. QA targets are generated programmatically and verified against t^{vis} (see section 4), so the supervision remains grounded in the renderer's execution outputs.

Overall Objective. The full set of per-instance losses spans forward generation and reverse parsing:

$$\mathcal{L} = \underbrace{L_{\text{NL2Chart}}}_{\text{forward (generate)}} + \underbrace{L_{\text{Schema}} + L_{\text{Data}} + L_{\text{QA}}}_{\text{reverse (parse)}}. \tag{5}$$

All four losses use token-level cross-entropy without additional weighting. In each iteration, a single task is sampled, 80% from the three consistency tasks (NL2Chart, Schema Parsing, Data Parsing) and 20% from ChartQA. Only the corresponding loss term is computed.

The key design principle is *per-instance alignment*: all tasks for a given data instance share the same chart image and specification, so the forward generation and reverse parsing are always grounded in the same visual artifact. This is what distinguishes CycleChart from standard multi-task fine-tuning, where samples for different tasks are drawn independently and this alignment is lost. The consistency operates at the *data level*. The deterministic renderer bridges the forward and reverse paths by producing exact supervision targets from the generated specification, rather than requiring a differentiable round-trip loss as in classical cycle-consistency formulations for continuous domains. This simplicity is a feature: because every reverse-path target is an exact execution output with no annotation gap, the model receives a noise-free consistency signal that scales trivially with data, requires no reward modeling, and converges in as few as 400 steps (Figure 4).

6 Experiments

We evaluate CycleChart as a unified framework for chart generation, parsing, and reasoning. We conduct evaluations on CycleChart-Bench and several external benchmarks, compare against strong VLMs and chart-specialized models, and perform ablations across different backbones and consistency schedules.

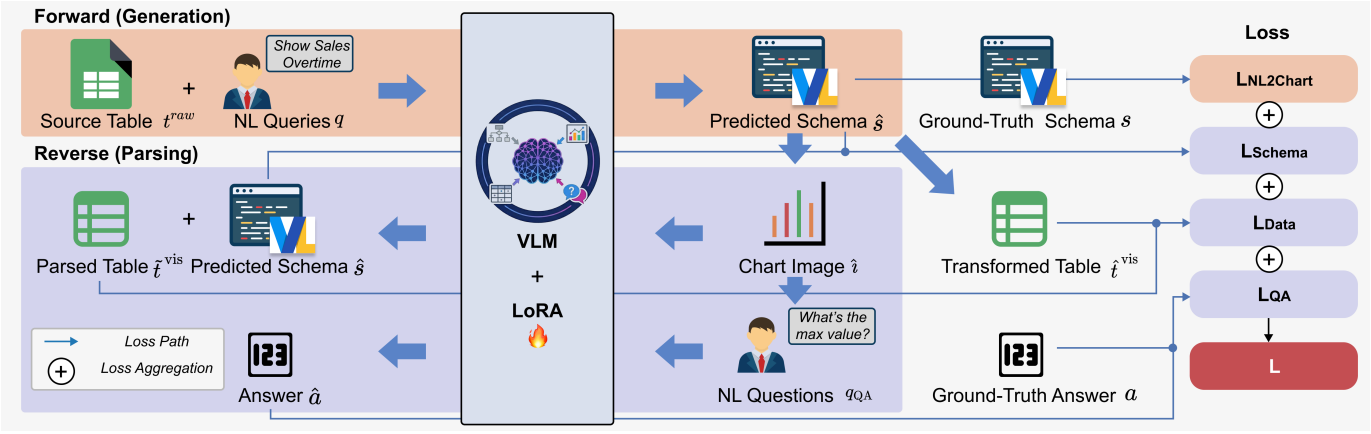

Fig. 3: Overview of the CycleChart training framework. Given a source table and an NL query, the model generates a chart specification (NL2Chart), which is executed to obtain a chart image and its visualization-level table. The model then parses the generated chart into a schema and table (Chart Parsing) and answers NL questions about the chart (ChartQA). All tasks use cross-entropy supervision, forming a closed generate–parse cycle that enforces consistency.

6.1 Implementation Details

We implement CycleChart on top of an instruction-tuned vision-language backbone. All models are trained with the same tokenizer and visual encoder as the underlying backbone. During training, each sample triggers one forward pass that computes the losses for the corresponding task (NL2Chart, Schema Parsing, Data Parsing, or ChartQA), depending on the sampling strategy described in section 5.

Vega-Lite rendering and data extraction. We adopt Altair as a front-end to the Vega-Lite compiler. For every predicted or ground-truth specification, we validate syntactic correctness and, if renderable, compile it into a visualization. Visualization-level tables t^{vis} are extracted directly from the Vega-Lite runtime, ensuring that data parsing is supervised with the exact values encoded in the chart marks.

Training setup. We fine-tune our models using AdamW with a constant learning rate of 1×10^{-5} and a per-device batch size of 2. LoRA-based parameter-efficient tuning (rank = 16, $\alpha = 32$, dropout = 0.05) is applied to both the *language model layers* and the *vision–language projector*, while the vision encoder and all other components remain frozen. We train for 2,000 steps with a constant learning-rate scheduler, allocating approximately 80% of the training steps to consistency-oriented tasks (NL2Chart, Schema Parsing, Data Parsing) and 20% to ChartQA. Training is performed on a single A100 GPU and we keep the total number of training steps constant in all ablations. All reported results are from a single training run; the low variance across our five-backbone ablation (Table 3), where every backbone improves consistently, suggests that the findings are robust.

Inference. We adopt a deterministic decoding setup for all evaluations to ensure fair and reproducible comparisons. Unless otherwise specified, inference uses greedy decoding (temperature = 0, top-k = 1, top-p = 1.0) with a maximum output length of 2,048 tokens.

6.2 Main Results

We use CycleChart-Bench as a unified benchmark to evaluate a wide range of models—including general-purpose VLMs and chart-specific models—and compare their performance against our CycleChart models. The goal is to assess how well existing models handle chart generation, parsing, and reasoning, and to quantify the advantage brought by our consistency-based training.

Baseline Selection. We select a diverse set of baseline models for comparison, including both general-purpose VLMs and chart-specialized models. The general-purpose VLMs include advanced commercial proprietary models like Gemini-2.5-Flash [3], Gemini-2.5-Pro [3], Claude-3.5-Sonnet [14], GLM-4.5v [8], and GPT-4.1 [15], which are known for their strong multimodal capabilities. The chart-specialized models include ChartInstruct-LLaMA2 [23], TinyChart-3B-768 [48], and unichart-chartqa-960 [22], which are specifically designed for chart-related tasks.

Metrics. We adopt task-specific metrics:

- **NL2Chart & Chart Schema Parsing.** Both produce a Vega-Lite specification, evaluated from three perspectives. *Textual similarity*: we compute ROUGE-L recall [17] on normalized (key-sorted) JSON strings to measure whether the generated specification covers the essential components of the ground truth. *Visual similarity*: we render both specifications as same-size images and compute PSNR [10] (pixel-level fidelity), MS-SSIM [34] (multi-scale structural consistency), and CLIP [28] (perceptual and semantic alignment); unrenderable specifications receive a zero score. *Validity*: the percentage of specifications that can be successfully rendered without errors.
- **Chart Data Parsing.** We use Relative Number Set Similarity (RNSS), following ChartQA [20], which measures the overlap between predicted and ground-truth data values while accounting for numerical scale. Although Relative Mapping Similarity (RMS) [48] is a stronger metric, our extracted tables cannot always be parsed as key-value mappings, so we adopt RNSS for generality. RNSS intentionally penalizes structural mismatches (e.g., wide vs. long format), which is appropriate because our Data Parsing task is *schema-conditioned*: deviations indicate a failure to follow the visual specification rather than a harmless format difference.
- **ChartQA task.** We use tolerance-aware Exact Match (EM): numeric predictions are accepted within a small absolute or relative threshold, while textual answers are matched after normalization.

Analysis. For NL2Chart (Table 1), CycleChart-7B achieves the best performance across all textual, visual, and validity metrics. The 3B variant performs strongly, showing that our consistency-based training reliably produces both textually correct and visually faithful specifications. Among general VLMs, Gemini-2.5-pro offers the strongest visual similarity, while Claude-3.5-sonnet, Gemini-2.5-flash, and GPT-4.1 achieve reasonable textual overlap and validity. Chart-specific open-source models (e.g., ChartInstruct, UniChart, TinyChart) fail to produce valid Vega-Lite specifications; as they are not designed for forward chart generation, we exclude them from NL2Chart reporting.

For chart schema parsing (Table 2), CycleChart-7B again achieves the best performance, with the 3B version obtaining a strong result, highlighting the benefits of our training pipeline. Apart from CycleChart, general VLMs such as Gemini-2.5-pro and Gemini-2.5-flash also perform reasonably well on this task. ChartInstruct-LLaMA2,

Table 1: Zero-shot performance on NL2Chart tasks. External models are directly evaluated without fine-tuning. Bold indicates the best performance, and underline indicates the second best.

Model	NL2Chart				
	ROUGE-L recall ↑	PSNR ↑	CLIP ↑	MS-SSIM ↑	Valid ↑
General VLMs					
claude-3.5-sonnet	0.7666	20.8410	0.6821	0.4495	79.60%
gemini-2.5-flash	0.7987	10.6326	0.6886	0.4282	84.51%
gemini-2.5-pro	0.7752	21.1006	0.7678	0.5097	89.42%
glm-4.5v	0.6894	10.3034	0.4727	0.2826	54.75%
gpt-4.1	0.7369	17.2544	0.6909	0.4576	80.83%
Our Models (Trained on CycleChart-Bench)					
CycleChart-3B	0.8062	35.4636	0.8739	0.6180	96.93%
CycleChart-7B	**0.8712**	**43.8004**	**0.9098**	**0.6759**	**98.93%**

Table 2: Zero-shot performance on Chart Parsing and ChartQA tasks. Bold and underline indicate the best and second-best performance.

Model	Chart Schema Parsing					Chart Data Parsing	ChartQA Task
	ROUGE-L recall ↑	PSNR ↑	CLIP ↑	MS-SSIM ↑	Valid ↑	RNSS ↑	EM ↑
General VLMs							
claude-3.5-sonnet	0.8894	25.1227	0.7994	0.6030	84.82%	69.9386	0.5271
gemini-2.5-flash	0.9509	33.7034	0.8844	0.6886	91.41%	72.5527	0.6713
gemini-2.5-pro	0.9455	54.5638	0.8975	0.7846	91.41%	70.8386	0.6713
glm-4.5v	0.8704	20.8042	0.5321	0.3970	56.29%	70.6999	0.6015
gpt-4.1	0.9291	23.1719	0.6871	0.5128	71.63%	75.5297	0.6744
Chart-Specialized Models							
ChartInstruct-Llama2	–	–	–	–	–	–	0.3968
TinyChart-3B-768	–	–	–	–	–	46.5612	0.3131
Our Models (Trained on CycleChart-Bench)							
CycleChart-3B	0.9443	77.2901	0.9813	0.9011	99.85%	90.8364	0.6589
CycleChart-7B	**0.9732**	**89.2574**	**0.9973**	**0.9745**	**100.00%**	**95.2947**	**0.7550**

UniChart-chartqa-960, and TinyChart once again fail to follow the parsing instructions and cannot produce valid schemas.

For chart data parsing (Table 2), we follow the pipeline described in section 5: at test time the model first predicts a schema from the chart image, then extracts the data table conditioned on its own predicted schema (not the gold one). CycleChart-7B achieves an RNSS score of 95.29, significantly outperforming all existing models. The 3B variant also attains a strong score of 90.84, demonstrating the effectiveness of our consistency-based training in enhancing data extraction capabilities. Given larger model sizes and higher training data quality, proprietary VLMs like Gemini-2.5-pro and Claude-3.5-sonnet perform better than open-source chart-specific models. ChartInstruct-LLaMA2 and unichart-chartqa-960 are unable to follow the parsing instructions effectively and thus fail to produce valid outputs for this task.

For the ChartQA task (Table 2), CycleChart-7B reaches 0.7550 EM, beyond the performance of proprietary VLMs such as GPT-4.1 and Gemini-2.5-Pro. The 3B variant achieves 0.6589 EM, slightly below GPT-4.1 and Gemini-2.5-Pro but clearly outperforming all open-source chart-specialized baselines. These results show that consistency-based training substantially enhances chart reasoning and enables CycleChart to achieve strong zero-shot performance across diverse chart question-answering scenarios.

6.3 Ablation Study

We further evaluate the generalizability of our generate–parse consistency objective beyond CycleChart-Bench. To avoid evaluating only on the dataset used for training, we conduct ablation studies exclusively on *external* benchmarks spanning chart QA, parsing, and NL2Chart tasks. This setup enables us to isolate the effect of consistency-based training and verify that the performance gains are not tied to dataset-specific patterns but reflect real improvements in multi-task chart understanding and reasoning. The results are summarized in Table 3.

Baseline Selection. Our main experiments use Qwen2.5-VL-7B-Instruct as the default backbone; accordingly, Qwen2.5-VL-3B/7B-Instruct serve as the primary baselines in this ablation. To further test generality, we additionally include LLaVA-OneVision-1.5-4B-Instruct (a different architecture) and the more recent Qwen3-VL-4B/8B-Instruct, which were released after our framework was developed. Together, the five backbones span two model families, three architectures, and capacities from 3B to 8B.

External Datasets and Metrics. Our evaluation spans three major task categories:

- **NL2Chart:** evaluated with **Rouge-L** on nvBench [19], measuring overlap between generated and reference VQL outputs.
- **Chart Data Parsing:** evaluated on chart-to-table examples from ChartMOE-Align [44] with **Relax Acc.**, measuring key alignment and cell-level correctness under mild numeric/textual tolerances.
- **ChartQA:** evaluated on ChartQA [20], ChartQA-Pro [21], and CharXiv [35] (QA only; schema parsing is not evaluated on CharXiv, as its charts are not Vega-Lite) with **Exact Match (EM)**, combining numeric and textual matching with minor tolerance.

Analysis. Consistency training yields improvements on every benchmark across all five backbones, confirming the generality of the generate–parse objective.

Cross-generation comparison at a similar scale. Comparing Qwen2.5-VL and Qwen3-VL at similar model sizes reveals complementary patterns. At the small scale (3B vs. 4B), both models obtain comparable deltas on Chart Data Parsing (+.032 vs. +.033) and the ChartQA benchmark (+.024 vs. +.037), while Qwen3-VL-4B benefits more on ChartQA-Pro (+.033 vs. +.024) and CharXiv (+.016 vs. +.009). At the large scale (7B vs. 8B), Qwen2.5-VL-7B gains more on NL2Chart (+.036 vs. +.012) and Chart Data Parsing (+.030 vs. +.005), whereas Qwen3-VL-8B shows substantially larger improvements on the ChartQA benchmark (+.098 vs. +.046), ChartQA-Pro (+.052 vs.

Tasks	NL2Chart		Chart Data Parsing		ChartQA Task					
Backbone	NVBench Rouge-L↑	+Consist. (Δ)	ChartMOE-Align Relax Acc.↑	+Consist. (Δ)	ChartQA EM↑	+Consist. (Δ)	ChartQA-Pro EM↑	+Consist. (Δ)	CharXiv EM↑	+Consist. (Δ)
LLaVA-OV-4B	.683	.691 (+.008)	.746	.748 (+.003)	.842	.843 (+.001)	.365	.371 (+.006)	.418	.436 (+.018)
Qwen2.5-VL-3B	.705	.717 (+.012)	.682	.714 (+.032)	.780	.804 (+.024)	.295	.319 (+.024)	.349	.358 (+.009)
Qwen2.5-VL-7B	.681	.717 (+.036)	.729	**.758** (+.030)	.800	**.846** (+.046)	.339	**.377** (+.039)	.371	.429 (+.058)
Qwen3-VL-4B	.701	.709 (+.008)	.718	.751 (+.033)	.774	.811 (+.037)	.307	.340 (+.033)	.428	.444 (+.016)
Qwen3-VL-8B	.711	**.723** (+.012)	.751	.756 (+.005)	.725	.824 (+.098)	.296	.348 (+.052)	.397	**.472** (+.075)

Table 3: **Ablation across backbones on external benchmarks (zero-shot).** Each backbone is evaluated as-is and with our generate–parse *consistency* training (+Consist.), trained only on **CycleChart-Bench** with no tuning on the target benchmark. Parenthesized values denote absolute improvement.

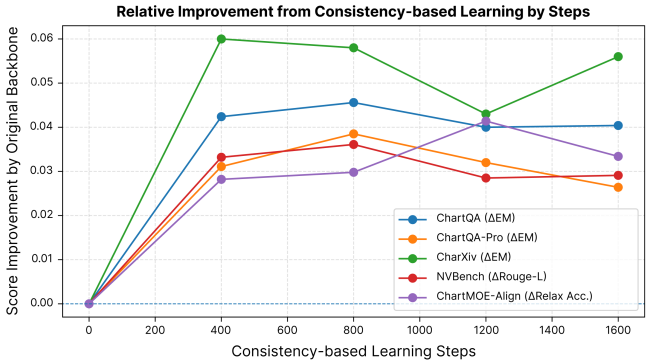

Fig. 4: **Impact of consistency training steps on external benchmarks.** Performance rises steeply within the first 400–800 steps and then plateaus, indicating that the generate–parse consistency objective is highly sample-efficient.

+.039), and CharXiv (+.075 vs. +.058). Notably, Qwen3-VL-8B starts from a lower ChartQA baseline than Qwen2.5-VL-7B (.725 vs. .800), yet consistency training closes much of this gap (+.098), suggesting that the cycle-consistent objective is especially effective at compensating for weaker chart-understanding priors.

Cross-architecture generalization. LLaVA-OneVision-4B, which uses a different vision encoder and LLM backbone, also benefits consistently, with its largest gain on CharXiv (+.018). The smaller absolute deltas compared to Qwen-family models are expected: LLaVA-OneVision already starts from a strong baseline (e.g., .842 on the ChartQA benchmark, the highest among all backbones before consistency training), leaving less headroom for improvement, a pattern consistent with the ceiling effect observed in subsection 6.5. Nonetheless, the consistent positive direction across all five benchmarks confirms that the consistency objective is architecture-agnostic and does not rely on Qwen-specific inductive biases.

CharXiv as a stress test. CharXiv contains charts from diverse scientific literature with visualization styles substantially different from the Vega-Lite charts used during training. Even so, all five backbones improve, with Qwen3-VL-8B reaching .472 (+.075), highlighting the strong out-of-distribution generalization of consistency-based learning.

6.4 Training Steps Ablation

We analyze how training steps of consistency learning influence performance across tasks. As shown in Figure 4, most benchmarks exhibit a steep improvement within the first 400–800 steps, after which the gains plateau or oscillate slightly. CharXiv shows the largest early boost, while the ChartQA benchmark, ChartQA-Pro, nvBench, and ChartMOE-Align all reach stable improvements in roughly the same range. This pattern indicates that CycleChart benefits quickly from consistency supervision: a relatively small number of steps is sufficient to align generation, parsing, and reasoning behaviors. Overall, the method achieves strong multi-task gains with modest training cost,

Table 4: Cycle-consistency vs. multi-task SFT on CycleChart-Bench. Both models use identical data, tasks, and training budget; only the per-sample alignment differs.

Task (Metric)	SFT-7B	CycleChart-7B	Δ
NL2Chart (ROUGE-L %)	84.75	**87.12**	+2.37
Schema Parsing (ROUGE-L %)	96.82	**97.32**	+0.50
Data Parsing (RNSS)	94.91	**95.29**	+0.38
ChartQA (EM)	72.09	**75.50**	+3.41

demonstrating the efficiency of consistency-based learning. This rapid convergence also makes early stopping practically important. Because CycleChart trains predominantly on Vega-Lite specifications, prolonged training risks overfitting to Vega-Lite-specific syntax and layout conventions. Early stopping at the plateau (400–800 steps) preserves the backbone's generalization ability, enabling the model to transfer to alternative output formats such as VQL, as demonstrated by the nvBench results in Table 3.

6.5 Cycle-Consistency vs. Multi-Task Fine-Tuning

A natural question is whether the gains of CycleChart stem from its cycle-consistency structure or simply from multi-task training on the same data. To isolate this factor, we compare CycleChart-7B with a **Multi-Task SFT** baseline that is trained on the *same* data and tasks but draws samples independently across tasks, breaking the per-sample alignment that underpins our generate–parse cycle. Concretely, Multi-Task SFT shuffles the four task pools (NL2Chart, Schema Parsing, Data Parsing, ChartQA) so that each mini-batch may contain schema parsing labels from one chart and QA labels from another, whereas CycleChart always draws all four tasks from the same sample.

As shown in Table 4, CycleChart-7B consistently outperforms Multi-Task SFT across all four tasks. The largest gain appears on the ChartQA task (+3.41 EM), suggesting that per-sample alignment between chart generation and parsing provides especially strong supervision for downstream reasoning. NL2Chart also benefits substantially (+2.37 ROUGE-L), indicating that parsing feedback improves generation quality. The more modest gains on Schema Parsing (+0.50) and Data Parsing (+0.38) are expected: both SFT and CycleChart already exceed 94 on these tasks, leaving little headroom; our data volume ablation (Table 5) similarly shows that structural tasks saturate early, confirming a ceiling effect rather than a weak signal. Importantly, the benefit of per-instance alignment is amplified on out-of-distribution data: as shown in Table 3, the same consistency objective yields up to +5.8% EM on CharXiv and +4.6% on the ChartQA benchmark, whose chart styles are entirely absent from training, whereas Multi-Task SFT, lacking per-instance alignment, cannot provide this form of structural generalization. These results confirm that CycleChart's advantage is not merely due to multi-task training, but specifically due to the cycle-consistent alignment that couples generation and parsing within each instance.

6.6 Data Volume Scaling

To understand how performance scales with dataset size, we train CycleChart-7B on 25%, 50%, 75%, and 100% of the CycleChart-Bench

Table 5: Performance at different data proportions on CycleChart-Bench. Each model is trained to convergence.

Task (Metric)	25%	50%	75%	100%
NL2Chart (ROUGE-L %)	87.34	89.54	91.18	**92.12**
Schema Parsing (ROUGE-L %)	97.48	97.44	98.26	**98.34**
Data Parsing (RNSS)	95.42	96.61	96.57	**96.82**
ChartQA (EM %)	77.20	80.62	83.56	**85.11**

Table 6: Single-view vs. faceted chart performance on CycleChart-Bench. "Base" denotes the Qwen2.5-VL-7B-Instruct backbone; "Ours" denotes CycleChart-7B.

	Single-View		Faceted	
Task (Metric)	Base→Ours	Gain	Base→Ours	Gain
NL2Chart (Valid %)	40.9→98.9	+58.0	19.3→99.2	**+79.9**
Schema (ROUGE-L %)	70.7→97.5	+26.8	64.3→96.4	**+32.1**
Data Parsing (RNSS)	68.5→95.5	+27.0	66.0→94.1	**+28.0**
ChartQA (EM %)	58.3→75.7	**+17.4**	59.1→74.5	+15.5

training set. Unlike the main results (Table 1, Table 2), which use a fixed budget of 2,000 steps, each variant here is trained to convergence to isolate the effect of data volume. The 100% converged model therefore slightly outperforms the fixed-step main model; we retain the fixed-step configuration as our primary result because it provides a fairer comparison across ablations.

Table 5 reveals two distinct scaling patterns. Structural tasks (Schema Parsing and Data Parsing) reach near-saturation performance at just 25% of the data, with gains of less than 1 point from 25% to 100%. This high data efficiency suggests that the cycle-consistency objective provides strong inductive bias for recovering chart structure, requiring relatively few examples to learn the underlying Vega-Lite grammar. In contrast, the ChartQA task exhibits steady improvement across all data proportions, rising from 77.20 to 85.11 EM, consistent with the greater diversity of reasoning patterns required for question answering. NL2Chart follows an intermediate trend, improving consistently but with diminishing marginal returns. These results indicate that consistency-based learning is inherently sample-efficient: CycleChart-Bench's 6,507 charts, which are orders of magnitude smaller than existing chart corpora such as ChartInstruct (191K) or UniChart (611K), are sufficient for structural tasks, and further data augmentation would primarily benefit reasoning-oriented tasks like ChartQA.

6.7 Single-View vs. Faceted Chart Analysis

CycleChart-Bench includes both single-view charts (5,372) and faceted multi-view charts (1,135). To assess whether cycle-consistency training benefits more complex layouts disproportionately, we report a breakdown of CycleChart-7B's performance on these two subsets, comparing against the Qwen2.5-VL-7B-Instruct backbone.

Table 6 shows that CycleChart delivers larger absolute gains on the harder faceted subset for three out of four tasks. The most striking difference is in NL2Chart validity, where the baseline achieves only 19.3% on faceted charts (vs. 40.9% on single-view), yet CycleChart raises both to near-perfect validity (99.2% and 98.9%). Schema Parsing and Data Parsing show a similar pattern: the baseline struggles more with faceted charts, but CycleChart closes the gap, with gains of +32.1 and +28.0 on faceted charts compared to +26.8 and +27.0 on single-view. The ChartQA task is the exception, where single-view gains (+17.4) slightly exceed faceted gains (+15.5), likely because single-view questions can directly benefit from improved generation and parsing, whereas faceted QA requires additional cross-panel reasoning that is not fully captured by the cycle objective alone. Overall, these results confirm that cycle-consistency is especially effective for compositionally complex charts, where the structural priors enforced by the generate–parse loop most directly reduce specification errors.

6.8 Qualitative Analysis

Figure 5 presents representative ChartQA cases from CharXiv [35]. Its figures, which include dense multi-panel layouts, heavily annotated scientific figures, and map-like demographic visualizations, are considerably more complex than those in our training corpus. While our training data primarily consists of simple line, bar charts, and faceted layouts rendered via Vega-Lite, CharXiv includes chart types entirely absent during training, making it a strong test of generalization.

Successful reasoning (A–D). Case A requires consistent axis alignment and careful comparison between visually similar curves across paired subplots. The baseline misreads the temporal alignment and reports 14:00, whereas CycleChart retrieves the correct time 20:00. This improvement likely stems from the schema parsing objective, which trains the model to attend to axis labels and tick positions as part of recovering the chart specification, thereby reinforcing temporal grounding. Case B involves a demographic visualization with subtle color variations, a chart type not seen during training. The baseline reports it as unanswerable, suggesting that it fails to recognize the visual encoding entirely. CycleChart correctly infers the gender category, demonstrating that the structural priors learned from Vega-Lite-based charts (e.g., color-to-category mappings) transfer to unseen visualization styles. Case C asks how many subplots satisfy a condition on the maximum of a dotted line. The baseline undercounts (6 vs. ground truth 8), likely due to inconsistent attention across panels. CycleChart correctly identifies all qualifying subplots, suggesting that training on faceted chart generation and parsing strengthens cross-panel awareness. Case D involves reading an isotonic calibration curve at a specific ranking position. The baseline produces a plausible but incorrect estimate (65000 vs. ground truth 60000), whereas CycleChart returns the correct value. This reflects that generate–parse consistency encourages precise visual grounding over heuristic interpolation, as the model must learn to recover exact data values during the data parsing task.

Remaining failure modes (E–G). Case E involves a boxplot where the median line visually resembles axis tick marks. Both models produce incorrect estimates (Base: 15, Ours: 6, GT: 10), with errors in opposite directions, suggesting that neither has learned a reliable representation of boxplot-specific glyphs. Since our training data does not contain boxplots and the consistency objectives do not include glyph-type-specific structural cues, the model lacks the inductive bias needed to disambiguate the median line from other horizontal guides. Case F shows both models exhibiting moderate deviation when estimating a continuous value from a dense chart (Base: 38, Ours: 39.0, GT: 40). CycleChart produces a closer estimate, indicating partial improvement in axis–mark alignment, but the residual error reveals a fundamental precision ceiling: when axis ticks are sparse relative to the data range, sub-tick interpolation remains unreliable for current multimodal LLMs. Case G requires determining whether scatter points lie outside contour regions, a region–point inclusion task that is fundamentally geometric rather than encoding-level. The baseline reports it as unanswerable, while CycleChart attempts an answer (5) but overcounts (GT: 3), suggesting that it has learned to engage with spatial queries but lacks the geometric reasoning precision to resolve boundary cases. This class of spatial reasoning, which involves containment, intersection, and proximity, lies outside the scope of our current generate–parse objectives and represents a complementary challenge for future chart understanding research.

7 Discussion

Our results demonstrate that generate–parse consistency is a powerful and general training principle for chart understanding. Below we discuss the current limitations and promising extensions along four axes: data efficiency, grammar coverage, evaluation methodology, and offline versus online consistency.

Data efficiency. CycleChart-Bench contains only 6,507 charts, which is orders of magnitude smaller than ChartInstruct (191K) or UniChart (611K), yet CycleChart consistently outperforms models trained on those larger corpora. This data efficiency is a direct consequence of the generate–parse consistency objective: because every

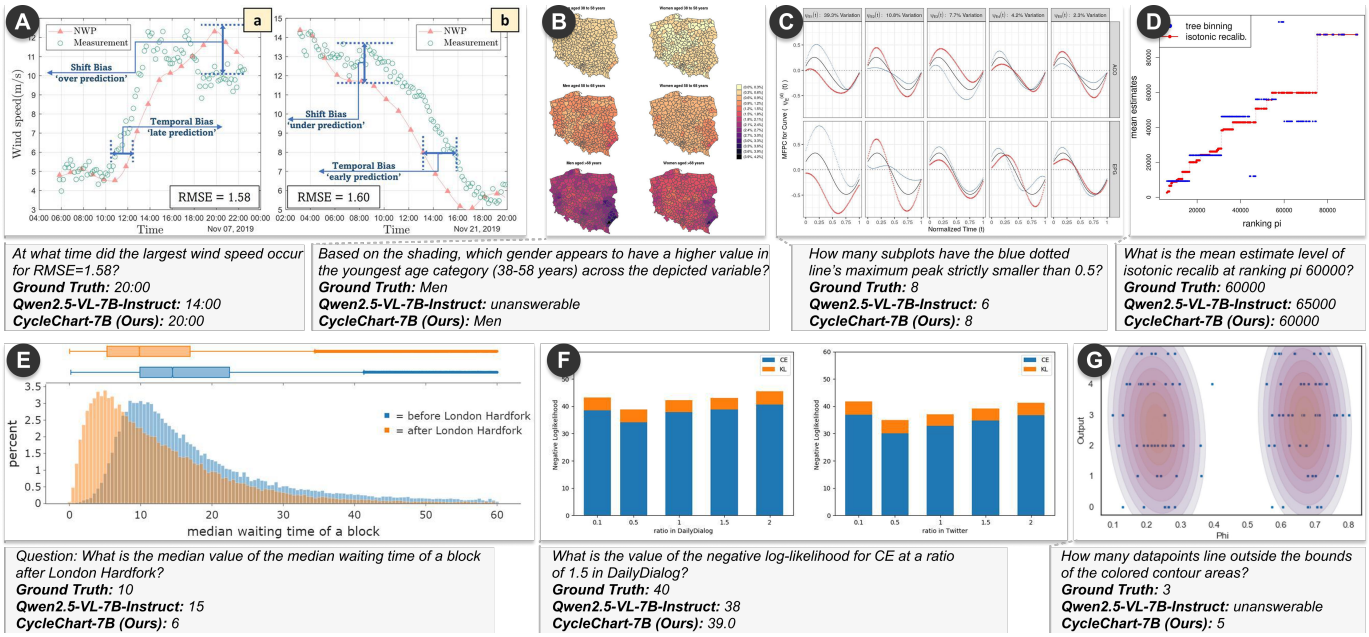

Fig. 5: Qualitative ChartQA cases from CharXiv [35], whose chart styles are absent from our training data. **Top row (A–D):** CycleChart-7B corrects baseline errors, demonstrating improved numerical grounding (A), categorical comparison in unseen chart types (B), counting across faceted subplots (C), and fine-grained value reading (D). **Bottom row (E–G):** Both models fail, revealing remaining challenges in glyph disambiguation for overlapping statistical marks (E), numerical drift on dense axes (F), and geometric region–point inclusion reasoning (G).

sample simultaneously supervises forward generation and reverse parsing through a shared instance, each chart provides far denser supervision than an independently sampled multi-task example. Our data scaling ablation (Table 5) further quantifies this effect: structural tasks saturate at just 25% of the data, whereas the ChartQA task improves steadily from 77.20 to 85.11 EM across the full range, suggesting that consistency-based learning is inherently sample-efficient for structural tasks and that reasoning-oriented tasks benefit most from additional data. A promising direction is to incorporate web-scraped or real-world charts with pseudo-specifications, which are generated by a strong VLM and verified through the cycle-consistency loop itself, to increase both visual and semantic diversity without requiring manual annotation. Furthermore, the current QA set is constructed from a fixed set of question templates; augmenting it with open-ended, compositional questions (e.g., multi-hop comparisons across chart panels) would better stress-test the reasoning capabilities of future models.

Visualization grammar coverage. The benchmark is restricted to Vega-Lite-style charts, creating a visual distribution gap relative to charts rendered by other libraries (e.g., matplotlib, D3, or hand-drawn). Our CharXiv transfer results (Table 3) show that the structural priors learned through cycle-consistency partially bridge this gap, as CycleChart-7B achieves +5.8% EM on CharXiv despite never seeing its chart styles. But generalization degrades for highly stylized or non-standard chart types such as infographics, 3D visualizations, and charts with heavy textual annotations. Crucially, the generate–parse consistency framework is grammar-agnostic: the cycle objective only requires a deterministic renderer R and a data extraction interface, both of which can be instantiated for any declarative grammar. Extending CycleChart to ECharts or matplotlib would therefore involve swapping the rendering and extraction components while retaining the same training loop, enabling a single model to develop consistency priors across multiple visualization ecosystems.

Automated visual evaluation. Our current evaluation relies on reference-based metrics (PSNR, CLIP, MS-SSIM) that require ground-truth renderings. While these metrics capture pixel-level fidelity and perceptual similarity, they are insensitive to semantically important but visually subtle differences. For example, a swapped legend color mapping produces a high PSNR but conveys entirely wrong information. An LLM-as-judge paradigm, where a strong VLM evaluates whether a rendered chart faithfully conveys the intended data and structure, could complement reference-based metrics with semantic correctness judgments. Such a judge could also serve as a reward signal for reinforcement-based fine-tuning, enabling self-improving consistency learning without pixel-level ground truth. This is particularly relevant for scaling to grammars where deterministic rendering is unavailable or where multiple valid visual encodings exist for the same data.

Offline versus online consistency. CycleChart enforces consistency offline: reverse-path targets are pre-computed from gold specifications rather than from the model's own predictions. This is deliberate—rendering every predicted $\hat{s}$ during training would introduce substantial I/O overhead and frequent failures from invalid specifications, especially early in training when validity is low (Table 6). The offline design produces noise-free targets, and per-instance alignment still ensures that forward and reverse tasks share the same data instance, which is the key property distinguishing CycleChart from standard multi-task training (Table 4). An online or hybrid schedule—switching to on-the-fly rendering once validity stabilizes—could tighten the cycle further and is a promising direction for future work.

8 Conclusion

We presented CycleChart, built on the observation that chart generation and understanding are two sides of the same coin: forcing a model to close the generate–parse loop on every instance yields far stronger representations than multi-task training on the same tasks independently. Our experiments surface three insights: *consistency over multi-tasking*—per-instance alignment, not task diversity, drives the gains; *renderer as free annotator*—deterministic Vega-Lite execution derives all reverse-path targets at zero labeling cost; and *compositionality as the bottleneck*—the cycle objective benefits compositionally complex faceted charts most, indicating that it teaches structure, not surface patterns. Together with strong transfer to unseen external benchmarks, these results position CycleChart and CycleChart-Bench as solid foundations for future chart understanding research.

REFERENCES

[1] A. Bansal, S. Ma, D. Ramanan, and Y. Sheikh. Recycle-gan: Unsupervised video retargeting. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 119–135, 2018. 2

[2] J. Chen, L. Kong, H. Wei, C. Liu, Z. Ge, L. Zhao et al. Onechart: Purify the chart structural extraction via one auxiliary token. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 147–155, 2024. 2

[3] G. Comanici, E. Bieber, M. Schaekermann, I. Pasupat, N. Sachdeva, I. Dhillon et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025. 5

[4] M. Cornia, L. Baraldi, H. R. Tavakoli, and R. Cucchiara. Towards cycle-consistent models for text and image retrieval. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pp. 0–0, 2018. 2

[5] D. Deng, A. Wu, H. Qu, and Y. Wu. DashBot: Insight-driven dashboard generation based on deep reinforcement learning. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):690–700, 2023. doi: 10.1109/TVCG.2022.3209468 2

[6] Y. Han, C. Zhang, X. Chen, X. Yang, Z. Wang, G. Yu et al. Chartllama: A multimodal llm for chart understanding and generation. *arXiv preprint arXiv:2311.16483*, 2023. 1, 2

[7] D. He, Y. Xia, T. Qin, L. Wang, N. Yu, T.-Y. Liu et al. Dual learning for machine translation. *Advances in neural information processing systems*, 29, 2016. 2

[8] W. Hong, W. Yu, X. Gu, G. Wang, G. Gan, H. Tang et al. Glm-4.5v and glm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning, 2025. 5

[9] E. Hoque and M. S. Islam. Natural language generation for visualizations: State of the art, challenges and future directions. In *Computer Graphics Forum*, vol. 44, p. e15266. Wiley Online Library, 2025. 2

[10] A. Horé and D. Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th International Conference on Pattern Recognition*, pp. 2366–2369, 2010. doi: 10.1109/ICPR.2010.579 5

[11] T.-Y. Hsu, C. L. Giles, and T.-H. Huang. Scicap: Generating captions for scientific figures. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 3258–3264, 2021. 2

[12] K.-H. Huang, H. P. Chan, Y. R. Fung, H. Qiu, M. Zhou, S. Joty et al. From pixels to insights: A survey on automatic chart understanding in the era of large foundation models. *IEEE Transactions on Knowledge and Data Engineering*, 2024. 1

[13] K.-H. Huang, M. Zhou, H. P. Chan, Y. Fung, Z. Wang, L. Zhang et al. Do LVLMs understand charts? analyzing and correcting factual errors in chart captioning. In L.-W. Ku, A. Martins, and V. Srikumar, eds., *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 730–749. Association for Computational Linguistics, Bangkok, Thailand, Aug. 2024. doi: 10.18653/v1/2024.findings-acl.41 1, 2

[14] A. Inc. Claude 3.5 sonnet news. https://www.anthropic.com/news/claude-3-5-sonnet, 2024. 5

[15] O. Inc. Introducing gpt-4.1 in the api. https://openai.com/index/gpt-4-1/, 2025. 5

[16] K. Kafle, B. Price, S. Cohen, and C. Kanan. Dvqa: Understanding data visualizations via question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5648–5656, 2018. 2

[17] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004. 5

[18] T. Luo, C. Huang, L. Shen, B. Li, S. Shen, W. Zeng et al. nvbench 2.0: A benchmark for natural language to visualization under ambiguity. *arXiv preprint arXiv:2503.12880*, 2025. 1, 2, 3

[19] Y. Luo, J. Tang, and G. Li. nvbench: A large-scale synthesized dataset for cross-domain natural language to visualization task. *arXiv preprint arXiv:2112.12926*, 2021. 1, 2, 6

[20] A. Masry, X. L. Do, J. Q. Tan, S. Joty, and E. Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the association for computational linguistics: ACL 2022*, pp. 2263–2279, 2022. 1, 2, 5, 6

[21] A. Masry, M. S. Islam, M. Ahmed, A. Bajaj, F. Kabir, A. Kartha et al. Chartqapro: A more diverse and challenging benchmark for chart question answering. *arXiv preprint arXiv:2504.05506*, 2025. 1, 2, 6

[22] A. Masry, P. Kavehzadeh, X. L. Do, E. Hoque, and S. Joty. Unichart: A universal vision-language pretrained model for chart comprehension and reasoning. *arXiv preprint arXiv:2305.14761*, 2023. 2, 5

[23] A. Masry, M. Shahmohammadi, M. R. Parvez, E. Hoque, and S. Joty. Chartinstruct: Instruction tuning for chart comprehension and reasoning. *arXiv preprint arXiv:2403.09028*, 2024. 2, 5

[24] F. Meng, W. Shao, Q. Lu, P. Gao, K. Zhang, Y. Qiao et al. Chartassistant: A universal chart multimodal language model via chart-to-table pre-training and multitask instruction tuning. *arXiv preprint arXiv:2401.02384*, 2024. 1, 2

[25] N. Methani, P. Ganguly, M. M. Khapra, and P. Kumar. Plotqa: Reasoning over scientific plots. In *Proceedings of the ieee/cvf winter conference on applications of computer vision*, pp. 1527–1536, 2020. 2

[26] A. Narechania, A. Srinivasan, and J. Stasko. 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, 2020. 1, 2

[27] J. Poco and J. Heer. Reverse-engineering visualizations: Recovering visual encodings from chart images. In *Computer graphics forum*, vol. 36, pp. 353–363. Wiley Online Library, 2017. 2

[28] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal et al. Learning transferable visual models from natural language supervision, 2021. 5

[29] Z. Shuai, B. Li, S. Yan, Y. Luo, and W. Yang. Deepvis: Bridging natural language and data visualization through step-wise reasoning. *arXiv preprint arXiv:2508.01700*, 2025. 1, 2

[30] B. Tang, A. Boggust, and A. Satyanarayan. Vistext: A benchmark for semantically rich chart captioning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7268–7298, 2023. 2

[31] Y. Tian, W. Cui, D. Deng, X. Yi, Y. Yang, H. Zhang et al. ChartGPT: Leveraging llms to generate charts from abstract natural language. *IEEE Transactions on Visualization and Computer Graphics*, 31(3):1731–1745, 2024. doi: 10.1109/TVCG.2024.3368621 1, 2

[32] A. Vogel, O. Moured, Y. Chen, J. Zhang, and R. Stiefelhagen. Refchartqa: Grounding visual answer on chart images through instruction tuning. In *International Conference on Document Analysis and Recognition*, pp. 523–537. Springer, 2025. 1, 2

[33] N. Wang, J. Deng, and M. Jia. Cycle-consistency learning for captioning and grounding. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24, art. no. 615, 9 pages. AAAI Press, 2024. doi: 10.1609/aaai.v38i6.28363 2

[34] Z. Wang, E. Simoncelli, and A. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, vol. 2, pp. 1398–1402 Vol.2, 2003. doi: 10.1109/ACSSC.2003.1292216 5

[35] Z. Wang, M. Xia, L. He, H. Chen, Y. Liu, R. Zhu et al. Charxiv: Charting gaps in realistic chart understanding in multimodal llms. *Advances in Neural Information Processing Systems*, 37:113569–113697, 2024. 1, 2, 6, 8, 9

[36] J. Wei, N. Xu, J. Zhu, G. Wu, Q. Chen, B. Yu et al. Chartmind: A comprehensive benchmark for complex real-world multimodal chart question answering. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 4555–4569, 2025. 2

[37] L. Wilkinson. The grammar of graphics. In *Handbook of computational statistics: Concepts and methods*, pp. 375–414. Springer, 2011. 1

[38] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE transactions on visualization and computer graphics*, 22(1):649–658, 2015. 2

[39] K. Wongsuphasawat, Z. Qu, D. Moritz, R. Chang, F. Ouk, A. Anand et al. Voyager 2: Augmenting visual analysis with partial view specifications. In *Proceedings of the 2017 chi conference on human factors in computing systems*, pp. 2648–2659, 2017. 2

[40] Y. Wu, L. Yan, L. Shen, Y. Wang, N. Tang, and Y. Luo. Chartinsights: Evaluating multimodal large language models for low-level chart question answering. *arXiv preprint arXiv:2405.07001*, 2024. 2

[41] R. Xia, H. Ye, X. Yan, Q. Liu, H. Zhou, Z. Chen et al. Chartx & chartvlm: A versatile benchmark and foundation model for complicated chart reasoning. *IEEE Transactions on Image Processing*, 2025. 1, 2

[42] Z. Xu, S. Du, Y. Qi, S. Lu, C. Xu, C. Yuan et al. Chartpoint: Guiding mllms with grounding reflection for chart reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 426–436, October 2025. 2

[43] Z. Xu, S. Du, Y. Qi, C. Xu, C. Yuan, and J. Guo. Chartbench: A benchmark for complex visual reasoning in charts. *arXiv preprint arXiv:2312.15915*, 2023. 2

[44] Z. Xu, B. Qu, Y. Qi, S. Du, C. Xu, C. Yuan et al. Chartmoe: Mixture of diversely aligned expert connector for chart understanding. *arXiv preprint arXiv:2409.03277*, 2024. 2, 6

[45] Y. Yang, Z. Zhang, Y. Hou, Z. Li, G. Liu, A. Payani et al. Effective training data synthesis for improving mllm chart understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2653–2663, 2025. 2

[46] Y. Ye, J. Hao, Y. Hou, Z. Wang, S. Xiao, Y. Luo et al. Generative ai for visualization: State of the art and future directions. *Visual Informatics*, 8(2):43–66, 2024. 2

[47] Z. Yi, H. Zhang, P. Tan, and M. Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *Proceedings of the IEEE international conference on computer vision*, pp. 2849–2857, 2017. 2

[48] L. Zhang, A. Hu, H. Xu, M. Yan, Y. Xu, Q. Jin et al. Tinychart: Efficient chart understanding with visual token merging and program-of-thoughts learning. *arXiv preprint arXiv:2404.16635*, 2024. 2, 5

[49] H. Zheng, S. Wang, C. Thomas, and L. Huang. Advancing chart question answering with robust chart component recognition. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 5741–5750. IEEE, 2025. 1

[50] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017. 2