CHARTANNO: Benchmarking Multimodal Large Language Models for Chart Annotation Generation

Zhenghan Chen, Zekai Shao, Lidan Tan, Xin Lin, Xingchen Zeng, Yi Shan, Ziyue Lin, Xiaoliang Fu, Xinyuan Liu, Yuetong Guo, Fen Wang, Bongshin Lee, and Siming Chen

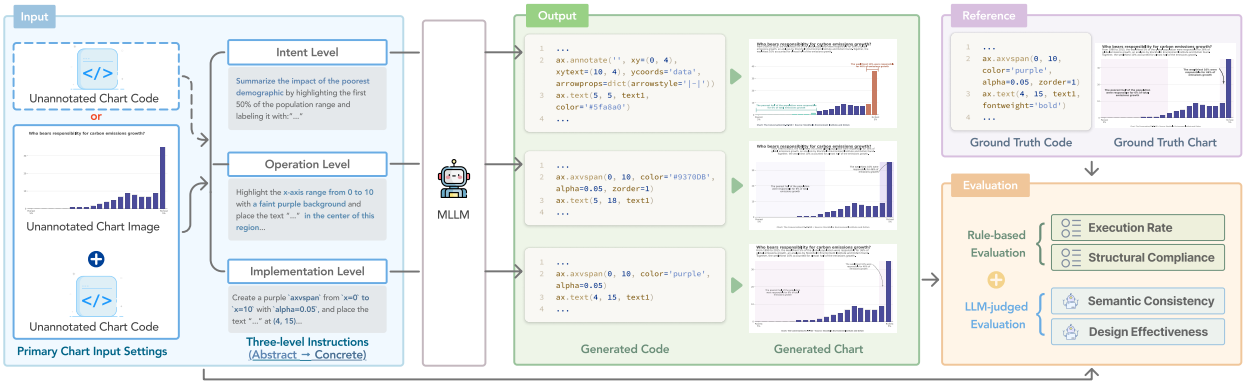


Figure 1: Task design of CHARTANNO. Inputs are Code or Code + Chart Image, paired with annotation instructions at three levels of specificity: abstract Intent, Operation, and concrete Implementation. For each chart input–instruction combination, the MLLM generates annotated code and a rendered chart, evaluated with reference to the unannotated code, instruction, and annotated ground truth.

Abstract—Annotations are essential to communicative visualization, helping explain data, emphasize key findings, and guide attention. While multimodal large language models (MLLMs) offer new opportunities for automatic chart annotation authoring, their capabilities in this task remain underexplored. To address this gap, we introduce CHARTANNO, a comprehensive benchmark for evaluating MLLMs on chart annotation generation. CHARTANNO contains 1,200 real-world charts with paired annotated and unannotated executable code, along with 3,600 annotation instructions spanning three levels of specificity. We also develop a multidimensional evaluation framework combining rule-based and LLM-judged metrics to assess execution, structural compliance, semantic consistency, and design effectiveness. We evaluate 10 representative MLLMs under two primary chart input settings: (1) chart code alone and (2) both code and chart image. Results reveal that proprietary models lead overall, though open-source models narrow the gap. While higher instruction specificity improves annotation quality, inferring abstract communicative intent remains difficult across all models. Providing chart images yields marginal benefit when code is available. We also examine the effect of chart code through an image-only ablation and analyze the effects of multiple task complexity indicators and instruction-level transitions. Further analyses characterize common failure modes and validate the reliability of the LLM-based judge. Experiments with D3 and SVG demonstrate the generalizability of CHARTANNO beyond its primary Python setting.

Index Terms—Chart annotation, benchmark, evaluation framework, multimodal large language models.

1 INTRODUCTION

Chart annotations are an important component of communicative visualization and data-driven storytelling, helping explain data, emphasize key findings, guide attention, and convey intended messages [39]. Widely used in real-world visualizations, annotations encompass diverse textual and graphical elements that augment existing charts [36, 37]. Their importance has motivated recent authoring systems and structured frameworks, including mixed-initiative systems for jointly authoring text and charts [41] and declarative approaches for specifying annotations [6, 38]. Despite these advances, creating effective annotations can still require manual effort.

Recent multimodal large language models (MLLMs) provide new opportunities to further support annotation authoring. Advances in MLLM capabilities [1, 10, 33, 34] have driven progress in chart understanding [29, 31, 52, 54, 62], generation [5, 47, 55, 58], and editing [18, 24, 60]. Chart annotation generation, however, presents a distinct challenge. Whereas chart editing often begins with a relatively explicit specification of the desired modification, annotation generation may begin with only a high-level communicative intent. A model must infer what information to convey, associate it with relevant chart elements, select an appropriate annotation design, and generate executable code that integrates the annotation without disrupting the existing chart.

Despite this challenge, systematic evaluation of MLLMs for chart annotation generation remains underexplored. Existing chart benchmarks largely focus on chart understanding, generation, or editing and rarely consider varying levels of instruction specificity in annotation authoring. Moreover, evaluating annotation generation requires accounting for multiple aspects of quality, including successful execution, structural

- *Zhenghan Chen, Zekai Shao, Lidan Tan, Yi Shan, Ziyue Lin, Xiaoliang Fu, Xinyuan Liu, Yuetong Guo, Fen Wang, and Siming Chen are with Fudan University. E-mail: {chenzh26, zkshao23}@m.fudan.edu.cn; simingchen@fudan.edu.cn.*
- *Xin Lin is with Sun Yat-sen University. E-mail: linx225@mail2.sysu.edu.cn.*
- *Xingchen Zeng is with The Hong Kong University of Science and Technology (Guangzhou). E-mail: xzeng159@connect.hkust-gz.edu.cn.*
- *Bongshin Lee is with the Department of Computational Science and Engineering and the Yonsei Institute of Digital Health, Yonsei University. E-mail: b.lee@yonsei.ac.kr.*
- *Zhenghan Chen and Zekai Shao are co-first authors, and Siming Chen and Bongshin Lee are co-corresponding authors.*

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

compliance, semantic consistency, and design effectiveness. These considerations motivate a benchmark that captures both instruction specificity and multidimensional annotation quality.

To address this need, we present ChartAnno,[1] a comprehensive benchmark for evaluating MLLMs on chart annotation generation. In chart authoring scenarios, annotations are often added to existing code-generated charts, making chart code a natural primary input representation for this task. Accordingly, as illustrated in Fig. 1, ChartAnno considers two primary chart input settings: (1) chart code alone and (2) chart code and the corresponding chart image. ChartAnno contains 1,200 public-facing and scientific charts curated from public chart datasets and arXiv papers, with manually refined chart representations and executable code. For each chart, we construct instructions at three levels of specificity: (1) Intent describes the communicative goal, (2) Operation specifies annotation actions and target relations, and (3) Implementation further provides concrete rendering parameters. Together, these yield 3,600 chart–instruction instances and 7,200 tasks across the two primary chart input settings. We additionally include an image-only setting with 3,600 tasks as an auxiliary ablation to assess model performance when chart code is unavailable.

We further develop an evaluation framework grounded in visualization knowledge. The framework combines rule-based analysis with LLM-based judgment across four complementary dimensions: *Execution Rate*, *Structural Compliance*, *Semantic Consistency*, and *Design Effectiveness*. The rule-based component adopts a differential analysis strategy, using the unannotated reference to assess preservation of the original chart and the difference between annotated and unannotated references to identify expected annotation elements for structured matching. For aspects that are difficult to capture with predefined rules, the LLM-based component evaluates whether the annotations faithfully communicate the intended information and whether their visual design effectively supports that communication.

Our evaluation with ChartAnno covers 10 recent MLLMs across three instruction levels, two primary chart input settings, and an auxiliary image-only ablation. Spanning both proprietary and open-source models, we evaluate GPT-5.4 [33], Gemini 3.1 Pro Preview [10], Gemini 3 Flash Preview [8], Claude Sonnet 4.6 [1], Kimi K2.5 [30], Gemma 4 31B [11], and four Qwen3.5 variants ranging from 9B to 397B parameters [34]. Our main results show that more detailed annotation instructions generally improve model performance, while Intent-level generation remains challenging. Providing chart images offers limited additional benefit when code is available. The Image-only ablation shows substantially lower performance, especially for open-source models and more detailed instructions. Performance also decreases systematically as chart and annotation complexity increase. We also validate our evaluation framework, finding strong agreement with human judgments and high consistency across repeated judging runs. Finally, we extend ChartAnno beyond Python to two additional representation formats, D3 and SVG, on a 120-chart subset, demonstrating the generalizability of our dataset and evaluation framework. The main trends observed with Python largely persist with D3 and SVG, while the representations exhibit performance trade-offs across instruction levels, suggesting that code length alone does not fully explain performance.

In summary, our main contributions are as follows:

- We formulate chart annotation generation as an MLLM benchmark task that spans from high-level communicative intents to concrete implementation specifications.
- We construct a high-quality and large-scale benchmark dataset of 1,200 real-world charts with paired unannotated and annotated executable code and 3,600 instructions at three levels of specificity, providing reusable resources for tasks such as chart generation.
- We introduce a multidimensional evaluation framework combining rule-based and LLM-judged metrics to assess execution, structural compliance, semantic consistency, and design effectiveness.
- We conduct a large-scale evaluation of 10 MLLMs across three instruction levels, two primary chart input settings, and an ablation setting, while validating the evaluation framework through strong human agreement and demonstrating its generalizability across different representation formats.

[1] Code and dataset are available at https://chartanno.github.io/.

2 Related Work

2.1 Chart Annotation

Annotation roles and design. Annotations have long been used as additional visual layers that augment charts with textual and graphical elements [21]. Prior work has systematically characterized how annotations are used in practice. Studies of real-world visualizations identify diverse annotation forms, targets, and communicative functions, and develop taxonomies and design spaces that capture common annotation practices [36, 37]. These studies show that annotations can combine textual and graphical elements, refer to different components of a visualization, and support different authoring goals. Other work examines structures formed by user-authored annotations [59] and the broader communicative functions of textual content in visualizations [42].

Annotation authoring and automation. Building on this design knowledge, visualization research has developed a range of tools and frameworks to support annotation creation. Contextifier automatically generates contextual annotations for stock visualizations [16], while ChartAccent supports interactive authoring of manual and data-driven annotations for data-driven storytelling [39]. Automatic graphical annotation generation has also been explored by deriving strategies from human annotation practices [40]. More recent approaches extend annotation authoring through mixed-initiative assistance and structured specifications. Pluto supports coordinated authoring of textual content and charts [41], while AnnoGram and ChartMark provide structured, grammar-based representations for specifying chart annotations [6, 38].

Annotations in communication and interpretation. Annotations play broader roles in visual analysis and communication, where they can externalize insights and connect visual evidence with narrative explanations. For example, annotations have been integrated into visual dashboards to support analytical exploration [3] and into data-driven storytelling systems to communicate findings alongside visualizations [17]. Empirical studies further show that the amount, semantic content, and placement of textual annotations can affect readers' preferences, takeaways, and interpretation of visualized information [43, 44]. An empirical study evaluates in-situ annotations generated using a single VLM on eight charts, finding improved accuracy for basic factual reading but no significant gains for simple interpretation or response time [14].

Despite substantial research on annotation design, authoring, and application, systematic evaluation of executable chart annotation generation by MLLMs remains underexplored. Building on these foundations, our work evaluates annotation generation across different levels of instruction specificity, considering whether generated annotations preserve the underlying chart while remaining structurally compliant, semantically faithful, and visually effective.

2.2 MLLMs for Chart Generation, Editing, and Annotation

Recent MLLMs have advanced chart generation and editing, motivating benchmarks of their ability to translate natural language, visual inputs, and existing chart representations into executable visualizations.

Chart generation. Natural-language-to-visualization research has progressed from NL-to-visualization mapping toward more advanced generation, evaluation, and agentic workflows. Early benchmarks such as nvBench [28] provide large-scale natural-language and visualization pairs, while nvBench 2.0 extends this setting with an ambiguity-aware benchmark, multiple valid visualizations, and reasoning paths [27]. ChartGPT constructs an instruction-chart dataset for fine-tuning LLMs to generate charts [47], while VisEval establishes visualization-specific criteria for assessing generated charts [5]. Text2Vis further extends text-to-visualization generation toward analytical tasks through an agentic workflow with automated evaluation [35].

A related line of work investigates chart-to-code generation, extending chart generation toward executable visualization programs. Plot2Code introduces image-to-code chart reconstruction [53], while ChartMimic and RealChart2Code extend this setting toward instruction-driven and real-world chart reconstruction scenarios [55, 58].

Chart2Code further considers broader chart authoring scenarios [45]. Recent approaches also investigate iterative refinement and agent-based workflows for chart authoring beyond one-shot generation [12, 20, 23, 57]. Together, these studies primarily focus on creating or reconstructing charts, whereas CHARTANNO investigates augmenting existing charts with communicative annotations.

Chart editing. Chart editing focuses on modifying an existing visualization according to user instructions and has recently been studied under increasingly diverse settings. ChartEdit evaluates code-based chart editing from natural-language instructions [60], while ChartM^3 combines textual instructions with visual indicators for fine-grained multimodal editing [56]. ChartEditBench studies grounded multi-turn editing across successive modifications [18]. ChartEditor instead considers image-based chart editing without access to the original chart code [4], while FigEdit focuses on structured editing of scientific charts [24]. Recent work also explores end-to-end chart editing across both local visual changes and more global transformations [25].

CHARTANNO complements existing chart generation and chart editing benchmarks by systematically evaluating executable chart annotation generation. Beyond execution, it evaluates whether generated annotations preserve the underlying chart and are structurally compliant, semantically faithful, and visually effective. It further covers both public-facing and scientific charts. Some chart editing tasks partially overlap with annotation generation because they involve adding textual or graphical elements to existing visualizations. However, existing editing benchmarks generally assume that desired modifications are explicitly specified through editing instructions. In contrast, CHARTANNO considers instructions ranging from abstract communicative intent to concrete implementation, requiring models to infer appropriate annotations rather than simply apply predefined edits.

Chart annotation. Chart annotation generation remains largely unexplored in published benchmarks. During the preparation of this work, we became aware of a concurrent preprint, AnnoBench [49], which also studies chart annotation generation, but in a different setting. AnnoBench contains 342 charts across six visualization representations, two instruction levels, and multiple chart-description conditions. It uses reference-free LLM judging as its automated evaluation and reports experiments on sampled subsets, with inconsistent alignment between LLM and human judgments across settings. Among them, the annotation tasks associated with 58 professional charts are derived from the original real-world charts, whereas those for the remaining 284 Vega/Vega-Lite charts are constructed through an LLM-assisted pipeline rather than derived from existing real-world annotations. In comparison, CHARTANNO contains 1,200 real-world charts with paired executable references, three instruction levels, two primary chart input settings (Code and Code + Image), and an Image-only ablation, enabling large-scale evaluation across the full benchmark with rule-based and human-aligned LLM-judged metrics in Python, with further extensions to D3 and SVG. We examined whether a controlled comparison could be conducted using AnnoBench's 58 professional charts, whose associated annotation tasks are closest to our setting. However, only 28 samples support a matched reference-based comparison at the Operation and Implementation levels, which we consider insufficient for a reliable benchmark-level comparison.

Tab. 1 summarizes the differences between CHARTANNO and representative benchmarks. The Supplementary Material further discusses differences from chart editing tasks and provides cross-benchmark comparison.

Table 1: Comparison of CHARTANNO with representative visualization generation, chart editing, and annotation benchmarks. $\sim$ indicates partial overlap or coverage. Diverse Sources denotes coverage of both public-facing and scientific charts. Real-world denotes charts collected from existing real-world visualizations rather than charts constructed within the benchmark. Hybrid Eval. denotes evaluation combining rule-based or programmatic metrics with model-based judgment. Vis.-grounded Eval. denotes criteria derived from visualization domain knowledge rather than generic execution, similarity, or model-based measures.

Benchmark	Anno. Gen.	Diverse Sources	Real-world	Hybrid Eval.	Vis.-grounded Eval.
VisEval	✗	✗	✗	✓	✓
ChartMimic	✗	✗	✓	✓	✗
RealChart2Code	✗	✗	✗	✓	✗
ChartEdit	$\sim$	✗	✓	✗	✗
ChartEditBench	$\sim$	✗	✗	✓	✗
FigEdit	$\sim$	✗	✗	✗	✗
AnnoBench	✓	✗	$\sim$	✗	✓
CHARTANNO	✓	✓	✓	✓	✓

3 CHARTANNO

We develop CHARTANNO to systematically evaluate MLLMs for chart annotation generation. We describe its task formulation, benchmark construction, and multidimensional evaluation framework. Detailed construction procedures, metric implementation, scoring rubrics, and evaluation prompt are provided in the Supplementary Material.

3.1 Task Formulation

Executable Visualization Environment. We instantiate CHARTANNO in a controlled static-chart environment using Python-based executable code. similar executable settings have been adopted by recent chart generation and editing benchmarks, including ChartMimic [55], ChartEdit [60], VisEval [5], MatPlotAgent [57], and Text2Vis [35], enabling reproducible rendering, direct code execution, and programmatic inspection of chart elements. We further construct D3 and SVG representations for a subset of CHARTANNO and evaluate them in Sec. 5 to validate generalizability beyond the Python setting.

Chart Input Settings and Task Objective. In chart authoring workflows, authors often refine an existing chart by editing its underlying code to add annotations that communicate a specific message more clearly. Accordingly, CHARTANNO primarily considers chart code as the input representation and examines whether the corresponding chart image provides complementary visual information. As illustrated in Fig. 1, we consider two primary chart input settings: (1) chart code alone and (2) chart code together with the corresponding chart image. We additionally include an image-only setting as an auxiliary ablation to assess model performance when chart code is unavailable.

Instruction Specificity. Prior work on chart annotation distinguishes communicative purposes, annotation forms and targets, and concrete visual properties [6, 39]. These distinctions motivate our three instruction levels of increasing specificity. (1) ***Intent-level instructions*** describe what should be communicated, including necessary information not inferable from the chart, while leaving the annotation strategy to the model. (2) ***Operation-level instructions*** additionally specify annotation operations, target chart elements, and placement relationships, while leaving concrete rendering parameters unspecified. (3) ***Implementation-level instructions*** further provide rendering parameters that can be directly translated into executable annotation code. Together, these levels progressively reduce the annotation design decisions left to the model.

3.2 Benchmark Construction

We construct CHARTANNO through a three-stage pipeline, as shown in Fig. 2. The pipeline consists of chart collection and filtering, reconstruction and annotation removal, and instruction generation.

Chart Collection and Filtering. To ensure diversity in chart types, annotation forms, and application contexts, we construct a large candidate pool from existing chart benchmarks, including ChartQAPro [29], CharXiv [52], ChartMimic [55], and MatPlotBench [57], as well as visualization studies focused on chart annotations and textual content [37, 42]. We further supplement these sources with figures from CC BY 4.0 arXiv papers [2] released between February 2025 and February 2026 and accepted at leading peer-reviewed venues. We use Semantic Scholar [19] to retrieve publication metadata and MinerU [50] to parse the PDFs and extract figures, following the pipeline used in ChartFI [51]. Together, these sources yield 113,014 candidate figures.

We then apply a two-stage MLLM-based screening procedure. The first stage identifies and removes non-chart figures, while the second scores the remaining charts based on chart completeness, information

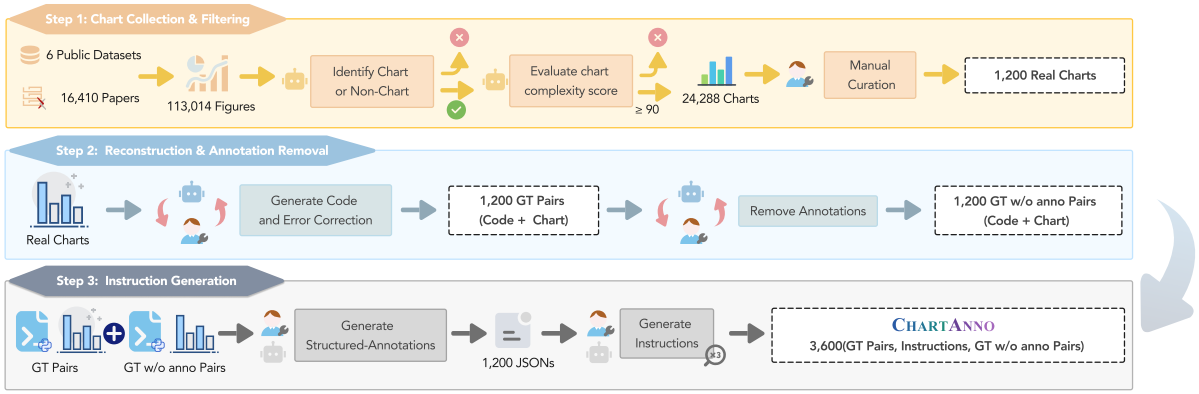


Figure 2: Construction pipeline of ChartAnno, including collection and filtering, reconstruction and annotation removal, and instruction generation.

richness, and visual complexity. Charts with scores below 90 are removed, leaving 24,288 candidates for manual review. Three authors further review the screened candidates for readability, information sufficiency, diversity in chart types and visual structures, and redundancy, while removing low-quality, incomplete, ambiguous, or overly simple charts and checking for obvious sensitive or inappropriate content. This process results in 1,200 annotated charts, including 653 public-facing and 547 scientific charts. As shown in Fig. 3, the dataset covers 17 chart types from seven data sources.

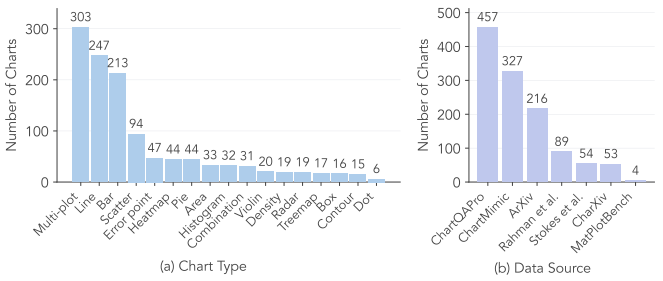


Figure 3: Distribution of ChartAnno by chart type and data source.

Chart Reconstruction and Annotation Removal. We reconstruct each selected chart in Python, obtaining an annotated ground truth (GT) pair (Code + Chart) consisting of executable code and its rendered image. We use LLMs (GPT-5.2 [32] and Gemini 3 Pro [9]) to generate initial reconstruction code for each selected chart. We evaluate reconstruction quality using a 100-point rubric adapted from ChartMimic [55], which assesses consistency with the source image in chart type, layout, text content, data, style, and clarity. The initial reconstructions achieve an average score of 92.34 on charts from sources other than ChartMimic. Based on this assessment, five authors manually refine the reconstructed charts for consistency with the source images and further correct three types of source-chart issues: (1) *ambiguous annotation intent*, where the intended target, direction, or connection of an annotation is unclear or inconsistent with the chart content; (2) *factual inconsistencies*, where annotation text conflicts with the underlying data or derived statistics, such as incorrect percentage changes or summary values; and (3) *visual presentation issues*, where annotations are difficult to interpret because of truncated text, occluded labels, or unclear placement. To construct the corresponding unannotated references, we categorize annotation elements by both annotation type and information source following prior annotation design-space research [37]. These labels guide LLMs in identifying and removing annotation-specific code, and five authors manually review and refine the outputs to produce the final unannotated GT pairs (Code + Chart), which preserve the underlying chart content and visual structure.

Instruction Generation. Public chart datasets generally do not provide the annotation rationales or original user requests that motivated

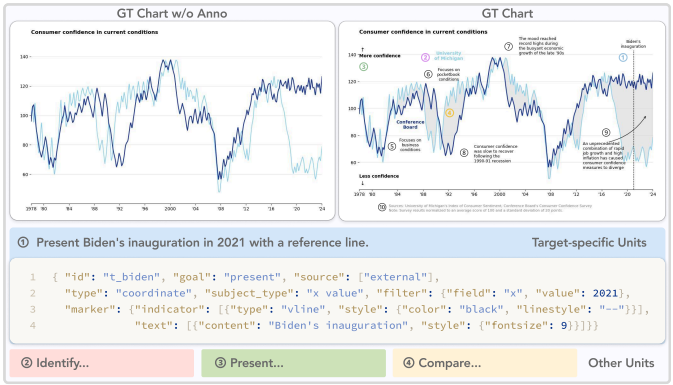


Figure 4: Example of the structured annotation representation with target-specific units. One unit is expanded in JSON.

individual annotations. Rather than attempting to recover their exact original wording, we aim to capture the underlying communicative intent and task requirements expressed by the annotations. Guided by prior work on chart annotation design spaces [37] and annotation grammars [6, 38], we represent the annotations in each chart using a structured schema for instruction construction, recording their communication goal (identify, compare, summarize, or present), target, source, content, annotation type, and markers [37]. Annotations associated with different targets are represented as separate target-specific units, as shown in Fig. 4. Based on these structured representations, we construct three instruction levels that progressively describe the annotation from communicative intent to annotation operations and concrete implementation details, as formally defined in Sec. 3.1.

We first prompt Gemini 3 Pro [9] to generate an initial structured annotation representation for each chart from the annotated and unannotated GT pairs. Three authors then review and refine these representations assisted by GPT-5.2 [32]. Based on the representations, we follow the same procedure to generate the three-level instructions, with Gemini 3 Pro producing the initial drafts and GPT-5.2 assisting in their refinement, yielding 3,600 instructions in total.

Dataset Statistics. ChartAnno contains 1,200 annotated GT pairs, 1,200 corresponding unannotated GT pairs, and 3,600 instructions. Across the GT charts, we identify 25,772 annotation elements, averaging 21.48 elements and 2.17 annotation types per chart. Fig. 5 summarizes the code and instruction length distributions. Using the Llama 2 tokenizer [48], annotated GT code has a median length of 1,349 tokens, compared with 865 tokens for unannotated GT code, with annotations adding a median of 363 tokens. Instruction length increases with specificity, with median lengths of 50, 76, and 84 words for the Intent, Operation, and Implementation levels, respectively, with the Intent-level median comparable to user instructions reported in C^2 [20]. The annotation extraction procedure is detailed in Sec. 3.3.1.

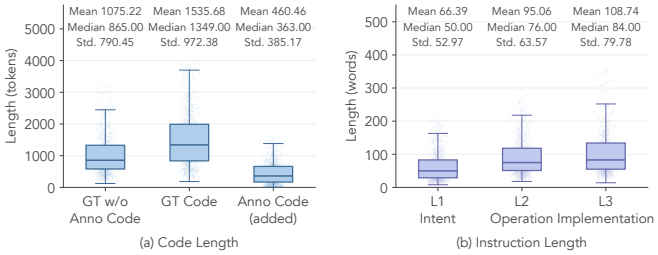


Figure 5: Code and instruction length distributions in CHARTANNO.

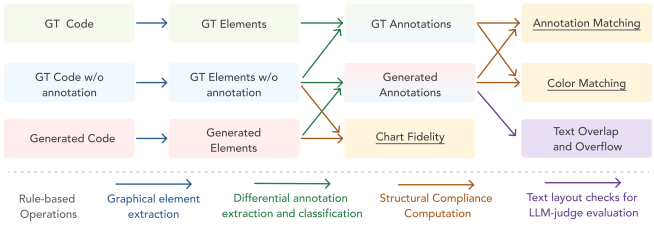


Figure 6: Workflow for computing *Structural Compliance* in CHARTANNO.

3.3 Evaluation Framework

Evaluating executable chart annotation generation requires both directly verifiable constraints and higher-level judgments of semantic and design quality. Low-level automatic metrics provide reproducible checks of properties such as executability and structural correctness, but cannot fully capture semantic correctness or overall design quality; visually similar chart outputs may still contain incorrect or unintended transformations [24]. Conversely, model- or human-based judgments can assess semantic alignment and visual quality more holistically, but may introduce evaluator variability and subjective bias [13]. Existing visualization benchmarks therefore combine complementary low- and high-level evaluation signals [5, 55, 60]. Following this strategy, we use rule-based metrics for properties that can be explicitly verified from rendered outputs and LLM judgment for qualities that require contextual interpretation of annotation meaning and visual design.

Accordingly, we evaluate annotations along four dimensions: *Execution Rate* and *Structural Compliance* (rule-based metrics), and *Semantic Consistency* and *Design Effectiveness* (LLM-judged metrics).

3.3.1 Rule-based Metrics

Rule-based evaluation focuses on properties that can be explicitly verified from executable and rendered chart outputs. Prior chart-generation benchmarks assess executability and rendered properties such as text, layout, chart type, and color [5, 55], while chart-editing benchmarks further consider preservation of unmodified content and correctness of graphical and textual changes [4, 18, 24, 60]. Building on these evaluations, we assess execution validity, preservation of the underlying chart, required annotation elements, and their visual properties, through annotation-specific graphical-element comparisons between paired annotated and unannotated GT charts.

Execution Rate. To measure whether the generated code executes and renders successfully, we assign 1 to instances that produce a chart without runtime errors and 0 otherwise. We report the proportion of successful instances, with failed executions receiving zero for all downstream quality metrics. We also summarize common execution failure modes to characterize model errors in executable chart generation.

Structural Compliance. For successfully rendered outputs, *Structural Compliance* evaluates whether annotation generation preserves the underlying chart and correctly realizes the specified annotation elements and their visual properties. For Operation- and Implementation-level instructions, we define *Structural Compliance* as the average of *Chart Fidelity*, *Annotation Matching*, and *Color Matching*. For Intent-level instructions, it consists only of *Chart Fidelity*, because the same communicative intent may be validly realized through different annotation structures and color designs. We operationalize these requirements by comparing rendered chart elements to assess chart preservation, annotation matching, and color matching. Fig. 6 summarizes the workflow based on generated code and annotated and unannotated GT code, with the computation procedure and individual metrics described below.

Graphical element extraction. Our evaluation operates on the graphical object representations underlying the rendered charts rather than source code, since equivalent visualizations may be implemented differently and expressed in different code representations; evaluating their graphical elements therefore reduces dependence on the underlying representation and supports evaluation across visualization languages. We first programmatically traverse the underlying graphical object hierarchy of the GT chart, GT chart w/o annotation, and generated chart to extract visible graphical and textual elements and their properties.

Chart Fidelity evaluates whether annotation generation preserves the original chart structure and data representation, which prior chart-editing work commonly assesses through human or model judgment [24, 60]. We instead operationalize preservation as a rule-based comparison between the generated chart and the GT chart w/o annotation, considering figure aspect ratio, axes layout and aspect settings, and the preservation of data-carrying marks and their encoded values. The metric is binary:

$$\text{ChartFidelity} = \begin{cases} 1, & \text{if all protected chart properties are preserved,} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Differential annotation extraction and classification. To identify newly introduced annotations, we perform element-level differencing using the GT chart w/o annotation as a common reference. We separately match elements in the GT chart and generated chart against those in the GT chart w/o annotation; unmatched elements are treated as newly introduced annotation candidates. Element matching is based on element type, textual content, spatial position and extent, shape, and data-related characteristics, with criteria adapted to different graphical elements. We classify the resulting candidates using heuristic rules into seven categories: enclosure, connector, text, glyph, color, indicator, and geometric annotations, following prior annotation taxonomies [36, 37].

Annotation Matching evaluates the correspondence between annotations in the generated and GT charts. We perform this comparison separately within each of the seven annotation categories. Unlike text annotations, which provide textual content as a relatively stable basis for correspondence, non-text annotations generally lack a unique element-level identity. The same annotation function may be realized with different graphical primitives, sizes, or placements (e.g., an enclosure highlighting the same chart region may vary in its exact extent and position while remaining semantically equivalent). We therefore adopt a relaxed matching criterion for non-text annotations, allowing variations in their exact graphical realization and assessing correspondence at the category level, where the intersection and union are defined as the minimum and maximum of the GT and generated annotation counts, respectively. For text annotations, content provides a stronger basis for correspondence, but equivalent text may still differ in representation (e.g., line wrapping, text-block splitting or merging, capitalization, or whitespace). We therefore normalize textual content and perform multiset matching, so that correspondence is determined primarily by text content rather than its exact rendered representation. Let I_k and U_k denote the resulting intersection and union counts for category k. *Annotation Matching* is computed using a Jaccard-style coefficient:

$$\text{AnnotationMatching} = \frac{\sum_k I_k}{\sum_k U_k}. \tag{2}$$

Color Matching evaluates how closely the colors of generated annotations match those of the GT annotations. We extract a primary color from each annotation element and perform matching separately within each annotation category, avoiding matches between colors that serve different annotation functions. For each GT–generated color pair that can be represented numerically, we convert both colors to the CIELAB

color space and compute their perceptual color difference ΔE_{00} using CIEDE2000 [26], while colors without a numeric representation are matched exactly. We convert the resulting difference into a normalized similarity:

$$s(c_i, c_j) = \max\left(0,\, 1 - \frac{\Delta E_{00}(c_i, c_j)}{100}\right). \quad (3)$$

Within each annotation category, we construct a pairwise color-similarity matrix and apply the Hungarian algorithm [22] to obtain the maximum-similarity one-to-one assignment between GT and generated colors. Let M denote the summed similarity of all matched pairs, and N_{GT} and N_{gen} denote the total numbers of GT and generated annotation colors, respectively. Precision, recall, and *Color Matching* are then computed as

$$P = \frac{M}{N_{\mathrm{gen}}}, \qquad R = \frac{M}{N_{\mathrm{GT}}}, \qquad \mathrm{ColorMatching} = \frac{2PR}{P+R}. \quad (4)$$

Text layout checks for LLM-judged evaluation. VisEval shows that fine-grained layout issues such as text overlap and overflow are difficult to assess reliably using GPT-4V alone and therefore supplements model-based readability assessment with explicit layout checks [5]. In chart annotation, however, overlap between non-text graphical elements is not necessarily undesirable, as annotations such as enclosures, highlights, and connectors may intentionally overlap existing chart content. We therefore restrict these geometric checks to newly added annotation text, computing text-overlap and off-canvas statistics as quantitative evidence for the subsequent LLM-based assessment of visual clarity rather than as standalone Structural Compliance scores.

3.3.2 LLM-Judged Metrics

Rule-based metrics capture executable and structural properties, but cannot assess whether annotations communicate the intended meaning or support effective visual design. We therefore develop an LLM-based evaluation rubric grounded in visualization knowledge on chart annotations. The judge evaluates the added annotations with reference to the instruction and annotated GT chart across five complementary aspects, aggregated into *Semantic Consistency* and *Design Effectiveness*.

Semantic Consistency. Chart annotation semantics involve multiple aspects, including association with relevant chart targets [39], contextual relevance [16], and diverse communicative purposes such as identifying, comparing, summarizing, and presenting information [37]. Beyond semantic correctness, annotations should also communicate their intended meaning clearly, as ambiguous annotation content can affect viewers' interpretations [36,43]. We define *Semantic Consistency* as the average of *Semantic Faithfulness* and *Semantic Clarity*.

Semantic Faithfulness evaluates whether the generated annotations correctly realize the meaning specified by the instruction. The judge considers whether annotations refer to the intended chart targets and accurately express required text, trends, values, relations, conclusions, and visual encodings, while checking for omissions, factual deviations, misreferences, and unsupported additions. A semantically correct annotation may still be difficult to interpret if its referent or relation to the chart is ambiguous. ***Semantic Clarity*** therefore evaluates whether annotation content forms a clear relation with the corresponding visual objects, such that the intended referent and meaning can be identified without competing interpretations or unnecessary inference.

Design Effectiveness. Beyond semantic quality, annotations should integrate with the underlying visualization and support its visual communication. Annotation research emphasizes readable placement, effective organization, and visual guidance without excessive clutter [36,37]. We define *Design Effectiveness* as the average of *Visual Clarity*, *Annotation Organization Quality*, and *Attention Guidance*.

Visual Clarity evaluates whether annotations remain readable without interfering with important chart content. The judge considers overlap, clipping, crowding, off-canvas placement, and occlusion of titles, axes, legends, labels, and data-carrying marks. The LLM judge receives both the rendered chart and quantitative rule-based evidence, including text-overlap and off-canvas statistics, and jointly considers them when assessing visual clarity. Annotations may remain readable while still being poorly arranged. ***Annotation Organization Quality*** evaluates how effectively annotation elements are placed, grouped, spaced, attached to their targets, and coordinated through color to form a coherent composition integrated with the underlying chart. It also considers redundancy, complexity, imbalance, and weak coordination among annotation elements. ***Attention Guidance*** evaluates whether annotations make the intended target or target set visually salient, establish a clear focus, and avoid competing or misleading emphasis that increases the effort required to locate the intended information.

Each submetric is scored on an integer scale from 1 to 5, with 0 for failed or missing cases. The same evaluation criteria and scoring rubrics are applied across all three instruction levels; only the role of the GT chart varies with instruction specificity. For Intent-level evaluation, the LLM judge is instructed to use the GT chart only to interpret the intended message and chart content, rather than as a reference for annotation designs. For Operation- and Implementation-level evaluation, the judge additionally references the GT chart when assessing compliance with the specified annotation requirements.

Table 2: Judge validation results. Human-rating reliability is measured by ICC, while judge–human and cross-judge consistency are measured by Spearman's ρ. Best results within each comparison group are **bolded**.

Comparison	Semantic Consistency	Design Effectiveness	Overall
Human 1 - Human 2 - Human 3	0.869	0.903	0.918
GPT-5.4–Human Avg.	**0.812**	**0.873**	**0.867**
Claude Sonnet 4.6–Human Avg.	0.808	0.805	0.832
Gemini 3.1 Pro–Human Avg.	0.714	0.732	0.748
GPT-5.4–Claude Sonnet 4.6	**0.730**	**0.783**	**0.789**
GPT-5.4–Gemini 3.1 Pro	0.716	0.772	0.782
Gemini 3.1 Pro–Claude Sonnet 4.6	0.664	0.652	0.668

Judge Validation and Selection. Before the large-scale evaluation, we conduct a pilot study on 90 randomly sampled outputs generated by Gemini 3 Flash Preview and independently rated by three coauthors to validate the rubric and select the LLM judge. Human ratings show high reliability, with ICCs of 0.869 for *Semantic Consistency*, 0.903 for *Design Effectiveness*, and 0.918 overall. Among GPT-5.4, Claude Sonnet 4.6, and Gemini 3.1 Pro Preview, GPT-5.4 shows the strongest alignment with aggregated human ratings, as shown in Tab. 2. Given that GPT-5.4 is also an evaluated model, we additionally examine cross-judge rank consistency and find strong correlations between GPT-5.4 and the other candidate judges. We therefore use GPT-5.4 as the judge for the large-scale evaluation.

4 Experiments

4.1 Experimental Setup

We use the proposed evaluation framework (Sec. 3.3) to evaluate various models with prompt protocols, as detailed below.

Models. We evaluate 10 MLLMs spanning proprietary and open-source models. The proprietary models include GPT-5.4 [33], Gemini 3.1 Pro Preview [10], Gemini 3 Flash Preview [8], and Claude Sonnet 4.6 [1]. The open-source models include Kimi K2.5 [30], Gemma 4 31B [11], and four Qwen3.5 variants (9B, 27B, 122B-A10B, and 397B-A17B) [34]. The proprietary models and Kimi K2.5 are accessed through APIs, while Gemma 4 31B and the Qwen3.5 models are locally deployed in BF16 precision on eight NVIDIA H100 80GB GPUs. We use deterministic decoding whenever supported, setting temperature to 0 and top-p to 1 where available.

Prompts. Following the chart input settings and instruction levels defined in Sec. 3.1, we use consistent prompt templates across all evaluated models. Under Code Input, models receive the chart code together with an Intent-, Operation-, or Implementation-level instruction and are asked to return executable Python code while preserving the content and style of the unannotated chart. For Code + Image Input, the chart image is additionally provided. For the Image-only ablation, we remove the chart code while retaining the chart image and instruction.

Table 3: Main evaluation results of 10 MLLMs under Code Input (left) and Code + Image Input (right) settings across three instruction levels (Intent, Operation, Implementation). Exec., Struct., Sem., and Design denote *Execution Rate*, *Structural Compliance*, *Semantic Consistency*, and *Design Effectiveness*, respectively. Gray Struct.* columns report *Chart Fidelity* only for Intent-level instructions and are not directly comparable with *Structural Compliance* at the other levels. Within each model category, best results are **bolded** and second-best results are underlined.

Model	Code Input												Code + Image Input											
	Intent-level				Operation-level				Implementation-level				Intent-level				Operation-level				Implementation-level			
	Exec.	Struct.*	Sem.	Design	Exec.	Struct.	Sem.	Design	Exec.	Struct.	Sem.	Design	Exec.	Struct.*	Sem.	Design	Exec.	Struct.	Sem.	Design	Exec.	Struct.	Sem.	Design
Proprietary Models																								
GPT-5.4	0.992	0.870	3.562	3.429	**0.996**	0.854	3.674	3.767	0.991	0.903	4.045	4.126	0.987	0.856	3.553	3.421	0.988	0.841	3.603	3.697	0.993	0.897	4.057	4.137
Gemini 3.1 Pro Preview	**0.994**	**0.922**	**3.688**	**3.641**	0.991	**0.873**	**3.777**	**3.907**	**0.998**	**0.919**	**4.128**	**4.206**	0.995	**0.917**	**3.680**	**3.639**	**0.993**	**0.876**	**3.780**	**3.909**	**0.997**	**0.916**	**4.148**	**4.230**
Gemini 3 Flash Preview	0.990	0.730	3.590	3.556	0.985	0.816	3.612	3.801	0.990	0.887	4.005	4.134	**0.996**	0.696	3.600	3.596	0.988	0.815	3.643	3.816	0.992	0.885	4.006	4.144
Claude Sonnet 4.6	0.975	0.864	3.493	3.385	0.963	0.828	3.461	3.571	0.984	0.903	3.999	4.098	0.984	0.885	3.512	3.421	0.974	0.843	3.498	3.646	0.988	0.906	4.020	4.112
Open-Source Models																								
Kimi K2.5	**0.973**	**0.860**	**3.315**	**3.267**	**0.964**	**0.812**	**3.307**	**3.422**	**0.969**	**0.877**	**3.869**	**3.959**	**0.983**	**0.877**	**3.310**	**3.280**	**0.965**	**0.818**	**3.333**	**3.455**	**0.972**	**0.878**	**3.878**	**3.961**
Gemma 4 31B	0.948	0.806	3.200	3.108	0.929	0.796	3.206	3.292	0.922	0.831	3.624	3.737	0.959	0.787	3.159	3.107	0.940	0.796	3.220	3.334	0.938	0.840	3.681	3.777
Qwen3.5-397B-A17B	0.933	0.802	3.082	3.012	0.905	0.752	3.000	3.086	0.948	0.841	3.617	3.736	0.953	0.832	3.035	2.988	0.945	0.780	3.051	3.201	0.945	0.838	3.656	3.770
Qwen3.5-122B-A10B	0.919	0.800	2.928	2.833	0.909	0.757	2.886	2.964	0.914	0.818	3.442	3.561	0.929	0.808	2.843	2.822	0.918	0.768	2.885	3.012	0.921	0.819	3.458	3.599
Qwen3.5-27B	0.935	0.828	2.966	2.896	0.896	0.750	2.830	2.934	0.904	0.813	3.432	3.552	0.939	0.828	2.959	2.900	0.894	0.746	2.837	2.960	0.914	0.821	3.440	3.550
Qwen3.5-9B	0.828	0.676	2.311	2.287	0.764	0.601	2.158	2.263	0.828	0.719	2.865	3.017	0.865	0.732	2.285	2.330	0.810	0.639	2.238	2.366	0.832	0.715	2.841	2.991

4.2 Results

Tab. 3 summarizes the performance of 10 MLLMs on ChartAnno. We first highlight the main findings and describe the effects of chart code, task complexity, and instruction specificity. We then report error analyses and assess the validity of the LLM-based judge.

4.2.1 Main Findings

Gemini 3.1 Pro Preview leads proprietary models, while Kimi K2.5 leads open-source models. Across three instruction levels, two chart input settings, and four metrics, Gemini 3.1 Pro Preview ranks first in 22 out of 24 comparisons within the proprietary group, with the only exceptions on *Execution Rate*. Among open-source models, Kimi K2.5 ranks first in all 24 comparisons. This shows that the leading models are consistently strong across execution, structure, semantics, and design, rather than excelling on a single metric.

A gap remains between proprietary and open-source models, but large open-source models narrow it. Proprietary models remain stronger overall, especially on *Semantic Consistency* and *Design Effectiveness*. However, Kimi K2.5 approaches proprietary models in some settings. For example, under Implementation-level Code Input, Kimi K2.5 reaches 3.869 in *Semantic Consistency* and 3.959 in *Design Effectiveness*, approaching Claude Sonnet 4.6's 3.999 and 4.098. At the Intent level, Gemini 3 Flash Preview shows relatively low *Structural Compliance*, falling below many evaluated open-source models. This suggests that strong semantic and design scores do not necessarily imply better preservation of the base chart, as analyzed in Sec. 4.2.3.

More detailed instructions improve annotation quality, with Intent-level generation being the most challenging. Performance generally improves as instructions become more detailed, especially in *Semantic Consistency* and *Design Effectiveness*. Although models achieve high *Execution Rate* under Intent-level instructions, their semantic and design scores remain lower than under Implementation-level instructions. For example, Gemini 3.1 Pro Preview with Code Input improves from 3.688 to 4.128 in *Semantic Consistency* and from 3.641 to 4.206 in *Design Effectiveness* from Intent to Implementation level. This indicates that current MLLMs can often produce executable code, but still struggle to infer suitable annotation targets and designs from abstract Intent-level instructions. See Sec. 4.2.2 for further analysis.

Chart image input brings marginal gains across evaluation metrics. As shown in Fig. 7, the normalized gain for any individual model and metric remains within 3%, with little change to the overall model ranking. Averaged across all models and instruction levels, the normalized gains are 0.85%, 0.49%, 0.11%, and 0.52% in *Execution Rate*, *Structural Compliance*, *Semantic Consistency*, and *Design Effectiveness*, respectively. All but *Semantic Consistency* show significant improvements (all $p < .001$), yet effect sizes are negligible (Cohen's g or $d_z < .10$). Image input has larger effects on rule-based metrics, especially for open-source models, while LLM-judged gains mainly appear in *Design Effectiveness*.

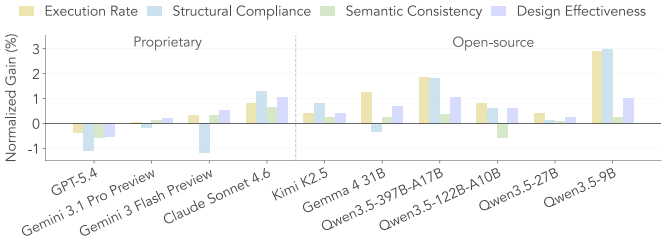


Figure 7: Normalized score gain from adding chart image input.

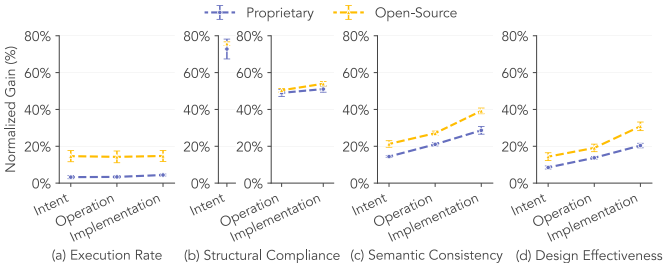


Figure 8: Normalized gains from adding chart code to Image-only input for proprietary and open-source models. Intent-level *Structural Compliance* is shown separately because it measures *Chart Fidelity* only.

4.2.2 Effects of Code, Complexity, and Instruction Specificity

Image-Only Ablation. To quantify the contribution of chart code, we compare the Image-only setting with the corresponding Code + Image setting. Overall, adding chart code substantially improves performance across evaluation metrics. As shown in Fig. 8, adding chart code yields larger gains for open-source models across most metrics, except for *Structural Compliance*, where the two groups show comparable gains. For the three quality metrics, gains generally increase with instruction specificity, with *Structural Compliance* showing this pattern from Operation to Implementation levels, whereas gains in *Execution Rate* remain relatively stable. *Structural Compliance* shows the largest gains overall. In particular, the substantial Intent-level gain, which reflects *Chart Fidelity* alone, highlights the difficulty of reconstructing the unannotated chart from image input alone. Together, these results show that chart code provides important grounding for annotation generation, particularly for open-source models and more detailed instructions. Complete Image-only results are provided in Supplementary Material.

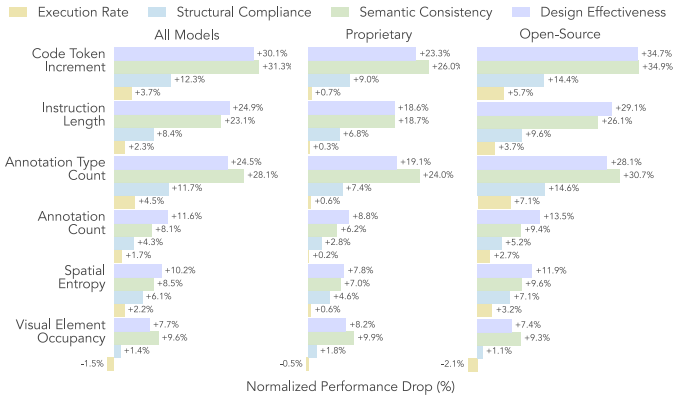


Figure 9: Normalized performance drops from simple to complex cases across complexity indicators and evaluation metrics.

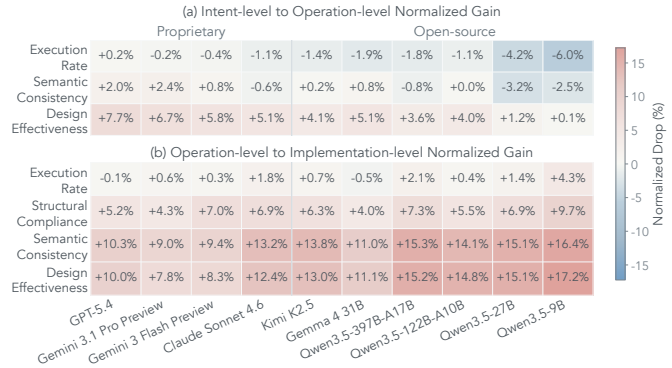


Figure 10: Normalized gains across instruction-level transitions. *Structural Compliance* is omitted from the Intent-to-Operation comparison because its Intent-level definition includes only *Chart Fidelity*.

Effect of Task Complexity. We examine how different aspects of complexity affect annotation generation using six indicators. *Code Token Increment* measures the additional code tokens introduced by annotations relative to the unannotated chart, reflecting the amount of code modification required. *Instruction Length* measures the amount of information provided to the model. *Annotation Type Count* records the number of distinct annotation types in a chart, while *Annotation Count* measures the total number of annotation elements. *Annotation Spatial Distribution Entropy* measures how broadly annotations are distributed across the chart, while *Visual Element Occupancy* measures the proportion of the chart area occupied by visual elements. Detailed definitions and computation procedures are provided in the Supplementary Material.

For each indicator, we sort charts by the corresponding value and split them into three equal-sized groups: simple, medium, and complex. We then compute the normalized performance drop from simple to complex cases. As shown in Fig. 9, larger code token increments correspond to the largest drops, especially in *Semantic Consistency* and *Design Effectiveness*. Higher annotation type counts and longer instructions also correspond to substantial performance declines, particularly in *Semantic Consistency* and *Design Effectiveness*. The drops are generally more pronounced for open-source models, especially for larger code token increments. These results show that ChartAnno exposes model limitations associated with both task complexity, such as larger code modifications and longer instructions, and annotation complexity, such as a greater number of annotation types. We also conduct complementary heatmap and regression analyses, which support these trends and are included in the Supplementary Material.

Effects of Instruction-Level Transitions. As reported in Sec. 4.2, performance generally improves as instructions become more detailed. We further examine how these improvements differ across the two instruction-level transitions. Fig. 10 shows distinct patterns across the two transitions. From Intent-level to Operation-level instructions, stronger models generally benefit more, with the largest gains in *Design Effectiveness*, whereas changes in *Semantic Consistency* and *Execution Rate* are smaller and even negative for some weaker models. In contrast, the transition from Operation-level to Implementation-level instructions yields more consistent gains, particularly in *Semantic Consistency* and *Design Effectiveness*, with larger improvements for weaker models. These patterns suggest that Operation-level instructions provide annotation strategies that stronger models are better able to translate into concrete implementations, whereas Implementation-level instructions specify annotation parameters more directly, reducing the implementation burden and thereby yielding larger gains for weaker models.

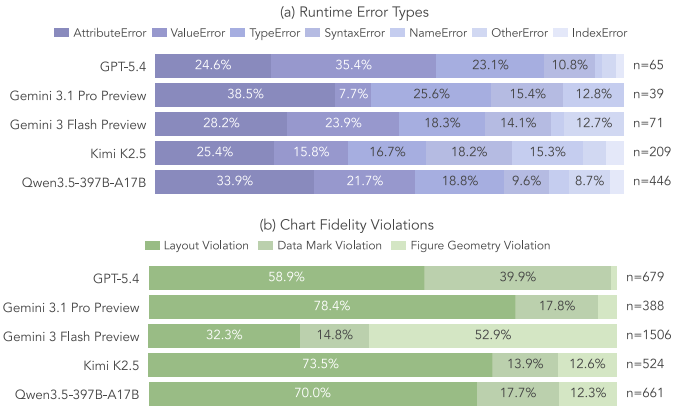


Figure 11: Error distributions for representative models. Bars show percentages; labels show counts.

4.2.3 Further Analyses

Error Analysis. We analyze model failures using distributional statistics and representative cases, focusing on two measurable binary failures: runtime errors and chart fidelity violations. As shown in Fig. 11, runtime errors are mainly caused by AttributeError, TypeError, and ValueError, suggesting failures in object references, value settings, and plotting API usage. For chart fidelity violations, layout, data mark, and figure geometry violations are common across models, with layout violations dominating in most cases. Gemini 3 Flash Preview is an exception, as its violations are dominated by figure geometry changes, which helps explain its low Intent-level *Structural Compliance* despite strong semantic and design scores in Sec. 4.2. Complete statistics and representative error cases are provided in Supplementary Material.

Validity of the LLM-based Judge. We further validate the GPT-5.4 judge on 600 randomly sampled outputs spanning models of different capability levels, including 300 outputs from Gemini 3.1 Pro Preview and 300 from weaker open-source models (150 each from Qwen3.5-9B and Qwen3.5-27B). The samples cover both primary input settings and all instruction levels. Three external visualization researchers (one PostDoc and two PhD students), each with at least two first-author IEEE TVCG publications, independently rated all 600 outputs. Each rater spent approximately 3.5 hours and received $50 in compensation; the human-evaluation protocol was approved by our institution's internal ethics review, and all raters provided informed consent. GPT-5.4 judgments are then compared with the human ratings and repeated runs. As shown in Tab. 4, GPT-5.4 shows strong overall alignment with human ratings and high repeat stability. Supplementary analyses further show consistently high correlations across instruction levels and for both stronger and weaker models.

To assess potential judge-model bias arising from GPT-5.4's dual role as an evaluated model and the primary judge, we conduct a cross-judge analysis following the cross-validation strategy used in Plot2Code [53]. The analysis covers 1,500 outputs from five models, including GPT-5.4 itself, evaluated by GPT-5.4, Claude Sonnet 4.6, and Gemini 3.1 Pro Preview. The three judges show strong overall consistency, with Cronbach's α values of 0.885, 0.898, and 0.913 for

Table 4: GPT-5.4 judge validation on 600 outputs. Human Alignment is measured by Spearman's ρ, and Repeat Stability by ICC(3,1).

Metric	Human Alignment	Repeat Stability
Semantic Consistency	0.8194	0.9109
Design Effectiveness	0.8378	0.9474
Overall	0.8593	0.9427

Semantic Consistency, *Design Effectiveness*, and Overall scores, respectively, and pairwise Spearman correlations ranging from 0.712 to 0.842 across the metrics. Moreover, removing GPT-5.4 from the judge set leaves the ranking of the five evaluated models unchanged, providing further evidence against substantial GPT-5.4-specific judging bias. Additional details are provided in Supplementary Material.

5 Generalization and External Validation

We further examine the generalizability of ChartAnno at both the dataset and evaluation, providing a basis for future extensions to additional visualization languages and representations. To this end, we evaluate two representative extensions, D3 and SVG, on a random 10% subset of ChartAnno (120 charts). For D3, we adopt a two-stage LLM-assisted conversion process. We first use GPT-5.4 to convert the GT Code w/o annotation from Python to D3, followed by manual refinement. We then construct the annotated D3 code by providing GPT-5.4 with both the unannotated D3 code and the annotated Python GT code. For SVG, we execute each D3 implementation and serialize the resulting SVG DOM. To reduce input tokens and LLM context burden, we simplify raw SVGs through structural compression, numerical precision reduction, and Ramer–Douglas–Peucker path simplification [7], while manually verifying that chart structure and annotation content are preserved, reducing token count by 53.6%. Overall, the average GT code token counts for D3 and SVG are 3.48× and 6.61× that of Python, respectively. For the instructions, we retain the Intent- and Operation-level instructions, while converting the Implementation-level instructions to match the target representation because they contain representation-dependent details.

To examine the generalizability of our evaluation framework, we apply the same framework to D3 and SVG, adapting Graphical Element Extraction and Differential Annotation Extraction and Classification in the rule-based evaluation (Sec. 3.3.1) to each representation. We evaluate all models on the 120-chart subset under both representations. Fig. 12 summarizes performance across representations and instruction levels. Detailed results are provided in Supplementary Material.

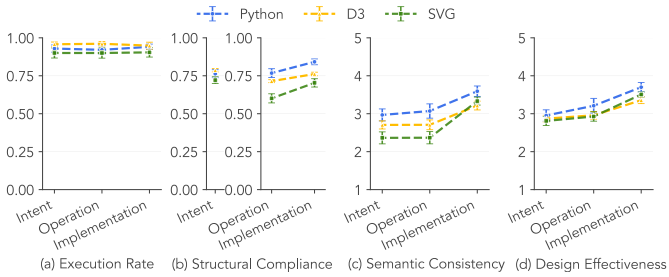


Figure 12: Average performance across Python, D3, and SVG representations over three instruction levels. Intent-level *Structural Compliance* is shown separately, as in Fig. 8.

Key performance trends under Python generalize to D3 and SVG. More specific instructions generally improve *Semantic Consistency* and *Design Effectiveness*, and the overall model ranking remains similar to that observed in the Python setting.

Representations exhibit different performance trade-offs across instruction levels. Model performance is generally highest with Python in *Semantic Consistency* and *Design Effectiveness*. For the rule-based metrics, models consistently achieve higher scores with D3 than with SVG across instruction levels, with smaller gaps at the Implementation level. For the LLM-judged metrics, average model scores are higher with D3 than with SVG at the Intent and Operation levels, but slightly higher with SVG at the Implementation level in *Semantic Consistency* (3.333 vs. 3.215) and *Design Effectiveness* (3.507 vs. 3.356). This reversal in average performance suggests that SVG benefits more from explicit Implementation-level guidance and that input length alone may not fully explain performance.

6 Discussion

Implications for Chart Annotation Authoring. Our findings suggest several directions for future MLLM-based annotation authoring systems. First, the difficulty of Intent-level generation indicates that current models still struggle to translate abstract communicative goals into appropriate annotation targets and designs. Interactive and mixed-initiative authoring can help users progressively articulate and refine such design intentions [39, 41]. Rather than relying on one-shot generation, future systems could support progressive specification and refinement, allowing users to move from high-level intent toward more explicit annotation operations or implementation details when needed. Extending prior annotation grammars such as AnnoGram and ChartMark [6, 38], our structured representation could support progressive refinement from abstract intents to concrete annotation specifications.

Second, the strong benefit of chart code and the limited additional gain from chart images suggest that executable chart representations provide semantic and structural grounding for annotation generation, while images offer complementary visual information. Future authoring systems could therefore combine code-based reasoning with visual feedback, using the former to preserve chart semantics and structure and the latter to support annotation layout and visual integration.

Finally, recent work has improved chart generation and editing through specialized data construction and post-training [4, 46, 61]. For chart annotation authoring, such improvement can be guided by evaluation signals that identify failures in conveying intended information, associating annotations with chart elements, and providing appropriate contextual and visual emphasis [16, 21, 39, 44]. Our multidimensional evaluation provides such signals across structural compliance, semantic communication, and visual design, supporting targeted regeneration and failure-focused data construction for post-training MLLMs.

Limitations and Future Work. Our primary evaluation focuses on Python-based static charts, while the D3 and SVG study on a benchmark subset demonstrates the generalizability of both the dataset and evaluation framework. Future work could extend ChartAnno to additional representations and interactive or animated charts. Although ChartAnno includes public-facing and scientific visualizations from diverse real-world sources, future work could further strengthen coverage of specific application domains through targeted sampling, including domains such as public health [15] and professional data journalism [49]. Finally, as MLLMs evolve, we will continue updating the ChartAnno GitHub leaderboard to track emerging models.

7 Conclusion

We introduced ChartAnno, a benchmark for evaluating MLLMs on chart annotation generation across three instruction levels and two primary chart input settings. Our evaluation of 10 MLLMs shows that proprietary models remain stronger overall, although large open-source models narrow the performance gap. More detailed instructions improve annotation quality, suggesting that models still struggle with abstract Intent-level generation. Providing chart images beyond chart code brings limited gains. The Image-only ablation performs substantially worse than the code-based settings. Additional analyses show that performance varies with multiple task complexity indicators and across instruction-level transitions. Further analyses characterize common failure modes and validate the reliability of the LLM-based judge. Experiments with D3 and SVG demonstrate the generalizability of ChartAnno. We hope ChartAnno can support research on more reliable chart annotation and visual communication.

REFERENCES

[1] Anthropic. Introducing Claude Sonnet 4.6. https://www.anthropic.com/news/claude-sonnet-4-6, 2026. Accessed: 2026-04-29. 1, 2, 6

[2] arXiv. arxiv. https://arxiv.org/, 2026. Accessed: 2026-05-22. 3

[3] S. K. Badam, S. Chandrasegaran, and N. Elmqvist. Integrating annotations into multidimensional visual dashboards. *Information Visualization*, 21(3):270–284, 2022. doi: 10.1177/14738716221079591 2

[4] L. Chen, Y. Xu, J. Ma, Y. Liu, D. Yang, L. Zhang et al. ChartEditor: A reinforcement learning framework for robust chart editing. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(24):20199–20207, 2026. doi: 10.1609/aaai.v40i24.39107 3, 5, 9

[5] N. Chen, Y. Zhang, J. Xu, K. Ren, and Y. Yang. VisEval: A benchmark for data visualization in the era of large language models. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):1301–1311, 2025. doi: 10.1109/TVCG.2024.3456320 1, 2, 3, 5, 6

[6] Y. Chen, Y. Wu, S. Shen, Y. Xie, L. Shen, H. Xiong et al. ChartMark: A structured grammar for chart annotation. In *2025 IEEE Visualization and Visual Analytics (VIS)*, pp. 311–315, 2025. doi: 10.1109/VIS60296.2025.00068 1, 2, 3, 4, 9

[7] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973. doi: 10.3138/FM57-6770-U75U-7727 9

[8] Google DeepMind. Gemini 3 Flash model card. https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Flash-Model-Card.pdf, 12 2025. Accessed: 2026-05-24. 2, 6

[9] Google DeepMind. Gemini 3 Pro model card. https://deepmind.google/models/model-cards/gemini-3-pro/, 2026. Accessed: 2026-08-26. 4

[10] Google DeepMind. Gemini 3.1 Pro model card. https://deepmind.google/models/model-cards/gemini-3-1-pro/, 2026. Accessed: 2026-04-29. 1, 2, 6

[11] Google DeepMind. Gemma 4 model card. https://ai.google.dev/gemma/docs/core/model_card_4, 2026. Accessed: 2026-04-29. 2, 6

[12] K. Goswami, P. Mathur, R. Rossi, and F. Dernoncourt. PlotGen: Multi-agent llm-based scientific data visualization via multimodal feedback. *arXiv preprint arXiv:2502.00988*, 2025. 3

[13] K. Goswami, P. Mathur, R. A. Rossi, F. Dernoncourt, V. Gupta, and D. Manocha. ChartEval: LLM-driven chart generation evaluation using scene graph parsing. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics: System Demonstrations*, pp. 86–93. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.ijcnlp-demo.10 5

[14] D. Haraguchi, K. Kikuchi, T. Suzuki, and N. Ogawa. Automating chart annotations for data storytelling: Does it enhance the efficiency and effectiveness of chart interpretation? In *Proceedings of the Extended Abstracts of the 2026 CHI Conference on Human Factors in Computing Systems*, CHI EA '26, art. no. 108, 7 pp. Association for Computing Machinery, New York, NY, USA, 2026. doi: 10.1145/3772363.3798871 2

[15] M. Hines and A. Ottley. Charting public health: A taxonomic study of visualization practices in the public health field. *arXiv preprint arXiv:2608.08657*, 2026. 9

[16] J. Hullman, N. Diakopoulos, and E. Adar. Contextifier: Automatic generation of annotated stock visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2707–2716, 2013. doi: 10.1145/2470654.2481374 2, 6, 9

[17] M. S. Islam, M. T. R. Laskar, M. R. Parvez, E. Hoque, and S. Joty. DataNarrative: Automated data-driven storytelling with visualizations and texts. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 19253–19286. Association for Computational Linguistics, Miami, Florida, USA, Nov. 2024. doi: 10.18653/v1/2024.emnlp-main.1073 2

[18] M. N. Kapadnis, L. Baghel, A. Naik, and C. Rosé. ChartEditBench: Evaluating grounded multi-turn chart editing in multimodal language models. *arXiv preprint arXiv:2602.15758*, 2026. 1, 3, 5

[19] R. Kinney, C. Anastasiades, R. Authur, I. Beltagy, J. Bragg, A. Buraczynski et al. The semantic scholar open data platform. 2025. 3

[20] W. Koh, J. Yoon, M. Lee, Y. Song, J. Cho, J. Kang et al. c^2: Scalable auto-feedback for LLM-based chart generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4525–4566. Association for Computational Linguistics, Albuquerque, New Mexico, 2025. doi: 10.18653/v1/2025.naacl-long.232 3, 4

[21] N. Kong and M. Agrawala. Graphical overlays: Using layered elements to aid chart reading. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2631–2638, 2012. doi: 10.1109/TVCG.2012.229 2, 9

[22] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, 1955. doi: 10.1002/nav.3800020109 6

[23] B. Li, Y. Wang, J. Gu, K.-W. Chang, and N. Peng. METAL: A multi-agent framework for chart generation with test-time scaling. In W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, eds., *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 30054–30069. Association for Computational Linguistics, Vienna, Austria, July 2025. doi: 10.18653/v1/2025.acl-long.1452 3

[24] L. Li, R. A. Rossi, S. Kim, S. Choudhary, F. Dernoncourt, P. Mathur et al. Charts are not images: On the challenges of scientific chart editing. In *The Fourteenth International Conference on Learning Representations*, 2026. 1, 3, 5

[25] S. Li, J. Sun, Z. Wang, X. Fan, H. Li, D. Yang et al. ChartE^3: A comprehensive benchmark for end-to-end chart editing. *arXiv preprint arXiv:2601.21694*, 2026. 3

[26] M. R. Luo, G. Cui, and B. Rigg. The development of the cie 2000 colour-difference formula: Ciede2000. *Color Research & Application*, 26(5):340–350, 2001. doi: 10.1002/col.1049 6

[27] T. Luo, C. Huang, L. Shen, B. Li, S. Shen, W. Zeng et al. nvbench 2.0: Resolving ambiguity in text-to-visualization through stepwise reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. 2

[28] Y. Luo, N. Tang, G. Li, C. Chai, W. Li, and X. Qin. Synthesizing natural language to visualization (nl2vis) benchmarks from nl2sql benchmarks. In *Proceedings of the 2021 International Conference on Management of Data*, SIGMOD '21, pp. 1235–1247. Association for Computing Machinery, New York, NY, USA, 2021. doi: 10.1145/3448016.3457261 2

[29] 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. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19123–19151. Association for Computational Linguistics, Vienna, Austria, July 2025. doi: 10.18653/v1/2025.findings-acl.978 1, 3

[30] Moonshot AI. Kimi K2.5: Visual agentic intelligence. https://www.kimi.com/blog/kimi-k2-5, 2026. Accessed: 2026-04-29. 2, 6

[31] K. Mukherjee, D. Ren, D. Moritz, and Y. Assogba. EncQA: Benchmarking vision-language models on visual encodings for charts. *IEEE Transactions on Visualization and Computer Graphics*, 32(1):648–658, 2026. doi: 10.1109/TVCG.2025.3634249 1

[32] OpenAI. Introducing GPT-5.2. https://openai.com/index/introducing-gpt-5-2/, 2025. Accessed: 2026-08-26. 4

[33] OpenAI. Introducing GPT-5.4. https://openai.com/index/introducing-gpt-5-4/, 2026. Accessed: 2026-04-29. 1, 2, 6, 3

[34] Qwen Team. Qwen3.5: Towards native multimodal agents. https://qwen.ai/blog?id=qwen3.5, 2026. Accessed: 2026-04-29. 1, 2, 6

[35] M. Rahman, M. T. R. Laskar, S. Joty, and E. Hoque. Text2Vis: A challenging and diverse benchmark for generating multimodal visualizations from text. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 31849–31874. Association for Computational Linguistics, Suzhou, China, Nov. 2025. doi: 10.18653/v1/2025.emnlp-main.1622 2, 3

[36] M. D. Rahman, B. Doppalapudi, G. J. Quadri, and P. Rosen. A survey on annotations in information visualization: Empirical studies, applications and challenges. *IEEE Transactions on Visualization and Computer Graphics*, 31(12):10439–10456, 2025. doi: 10.1109/TVCG.2025.3600957 1, 2, 5, 6

[37] M. D. Rahman, G. J. Quadri, B. Doppalapudi, D. A. Szafir, and P. Rosen. A qualitative analysis of common practices in annotations: A taxonomy and design space. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):360–370, 2025. doi: 10.1109/TVCG.2024.3456359 1, 2, 3, 4, 5, 6

[38] M. D. Rahman, M. Rahat-Uz-Zaman, A. McNutt, and P. Rosen. AnnoGram: An annotative grammar of graphics extension. In *2025 IEEE Visualization and Visual Analytics (VIS)*, pp. 236–240, 2025. doi: 10.1109/VIS60296.2025.00053 1, 2, 4, 9

[39] D. Ren, M. Brehmer, B. Lee, T. Höllerer, and E. K. Choe. ChartAccent: Annotation for data-driven storytelling. In *2017 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 230–239, 2017. doi: 10.1109/PACIFICVIS.2017.8031599 1, 2, 3, 6, 9

[40] D. Shi, A. Oulasvirta, T. Weinkauf, and N. Cao. Understanding and automating graphical annotations on animated scatterplots. In *2024 IEEE 17th Pacific Visualization Conference (PacificVis)*, pp. 212–221, 2024. doi: 10.1109/PacificVis60374.2024.00031 2

[41] A. Srinivasan, V. Setlur, and A. Satyanarayan. Pluto: Authoring semantically aligned text and charts for data-driven communication. In *Proceedings of the 30th International Conference on Intelligent User Interfaces*, 2025. doi: 10.1145/3708359.3712122 1, 2, 9

[42] C. Stokes, A. Arunkumar, M. A. Hearst, and L. Padilla. An analysis of text functions in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 32(1):769–779, 2026. doi: 10.1109/TVCG.2025.3634632 2, 3

[43] C. Stokes, C. X. Bearfield, and M. A. Hearst. The role of text in visualizations: How annotations shape perceptions of bias and influence predictions. *IEEE Transactions on Visualization and Computer Graphics*, 30(10):6787–6800, 2024. doi: 10.1109/TVCG.2023.3338451 2, 6

[44] C. Stokes, V. Setlur, B. Cogley, A. Satyanarayan, and M. A. Hearst. Striking a balance: Reader takeaways and preferences when integrating text and charts. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1233–1243, 2023. doi: 10.1109/TVCG.2022.3209383 2, 9

[45] J. Tang, H. H. Zhao, L. Wu, Z. Zhang, Y. Tao, D. Mao et al. From charts to code: A hierarchical benchmark for multimodal models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13467–13566. Association for Computational Linguistics, San Diego, California, United States, July 2026. doi: 10.18653/v1/2026.acl-long.616 3

[46] Z. Tang, X. Zhang, J. Yuan, Y. Zou, V. Gunjal, S. Jiang et al. Mm-recoder: Advancing chart-to-code generation with reinforcement learning and self-correction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22164–22173, 2026. 9

[47] 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, 2025. doi: 10.1109/TVCG.2024.3368621 1, 2

[48] H. Touvron, L. Martin, K. Stone, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 4

[49] M. R. uz Zaman, M. D. Rahman, A. McNutt, and P. Rosen. AnnoBench: A benchmark for visualization annotation generation. *arXiv preprint arXiv:2607.25911*, 2026. 3, 9, 2

[50] B. Wang, C. Xu, X. Zhao, L. Ouyang, F. Wu, Z. Zhao et al. MinerU: An open-source solution for precise document content extraction. *arXiv preprint arXiv:2409.18839*, 2024. 3

[51] F. Wang, Z. Shao, Q. Kang, C. Hu, Z. Zhang, L. Xie et al. ChartFI: Benchmarking faithfulness and insightfulness of chart descriptions from multimodal large language models. *arXiv preprint arXiv:2605.23694*, 2026. 3

[52] Z. Wang, M. Xia, L. He, H. Chen, Y. Liu, R. Zhu et al. Charxiv: Charting gaps in realistic chart understanding in multimodal LLMs. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. 1, 3

[53] C. Wu, Z. Liang, Y. Ge, Q. Guo, Z. Lu, J. Wang et al. 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*, pp. 3006–3028. Association for Computational Linguistics, Albuquerque, New Mexico, Apr. 2025. doi: 10.18653/v1/2025.findings-naacl.164 2, 8

[54] T. Xie, M. Lin, M. Liu, Y. Ye, C. Chen, and S. Liu. InfochartQA: A benchmark for multimodal question answering on infographic charts. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. 1

[55] C. Yang, C. Shi, Y. Liu, B. Shui, J. Wang, M. Jing et al. Chartmimic: Evaluating LMM's cross-modal reasoning capability via chart-to-code generation. In *The Thirteenth International Conference on Learning Representations*, 2025. 1, 2, 3, 4, 5

[56] D. Yang, L. Zhang, Z. Yue, L. Chen, Y. Xu, W. Wang et al. ChartM^3: Benchmarking chart editing with multimodal instructions. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 5001–5009, 2025. doi: 10.1145/3746027.3755714 3

[57] Z. Yang, Z. Zhou, S. Wang, X. Cong, X. Han, Y. Yan et al. MatPlotAgent: Method and evaluation for LLM-based agentic scientific data visualization. In L.-W. Ku, A. Martins, and V. Srikumar, eds., *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 11789–11804. Association for Computational Linguistics, Bangkok, Thailand, Aug. 2024. doi: 10.18653/v1/2024.findings-acl.701 3

[58] J. Zhang, Y. Li, Z. Li, X. Guo, J. Wu, L. Zheng et al. RealChart2Code: Bridging the gap in real-world chart-to-code generation via multi-task evaluation. In M. Liakata, V. P. Moreira, J. Zhang, and D. Jurgens, eds., *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 41995–42032. Association for Computational Linguistics, San Diego, California, United States, July 2026. doi: 10.18653/v1/2026.acl-long.1945 1, 2

[59] J. Zhao, M. Glueck, S. Breslav, F. Chevalier, and A. Khan. Annotation graphs: A graph-based visualization for meta-analysis of data based on user-authored annotations. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):261–270, 2017. doi: 10.1109/TVCG.2016.2598543 2

[60] X. Zhao, X. Liu, Y. Haoyue, X. Luo, F. Zeng, J. Li et al. ChartEdit: How far are MLLMs from automating chart analysis? evaluating MLLMs' capability via chart editing. In W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, eds., *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 3616–3630. Association for Computational Linguistics, Vienna, Austria, July 2025. doi: 10.18653/v1/2025.findings-acl.185 1, 3, 5

[61] X. Zhao, X. Luo, Q. Shi, C. Chen, S. Wang, Z. Liu et al. Chartcoder: Advancing multimodal large language model for chart-to-code generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7333–7348. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.3639

[62] Z. Zhu, M. Jia, Z. Zhang, L. Li, and M. Jiang. MultiChartQA: Benchmarking vision-language models on multi-chart problems. 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)*, pp. 11341–11359. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.566 1

SUPPLEMENTARY MATERIAL

A Comparison with Related Benchmarks **2**
A.1 Comparison with Chart Editing Tasks 2
A.2 Comparison with Annotation Tasks 2

B Data Construction Details **3**
B.1 Source Licenses and Intended Use 3
B.2 Manual Selection and Dataset Coverage 3
B.3 Reconstruction Quality and Correction Process 3

C Rule-based Evaluation Details **4**
C.1 Execution Environment 4
C.2 Rendered Element Representation 4
C.3 Annotation Element Extraction 4
C.4 Chart Fidelity 5
C.5 Annotation Matching 5
C.6 Color Matching 5
C.7 Text Relationship Statistics 6

D LLM-Judged Evaluation Rubric **6**
D.1 Semantic Faithfulness 6
D.2 Semantic Clarity 6
D.3 Visual Clarity 6
D.4 Annotation Organization Quality 6
D.5 Attention Guidance 7

E LLM-Judged Evaluation Prompt **7**
E.1 Evaluation Prompt Template 7
E.2 Instruction-Level Constraint Instantiations 8

F Generation Prompts and Examples **8**
F.1 Generation Prompt Templates 8
F.2 Prompt and Evaluation Example 8

G Model Configurations and Licenses **9**
G.1 Model Versions and Inference Settings 9
G.2 Computational Infrastructure and Budget 9
G.3 Model Licenses 9

H Chart Image Input Analysis **9**
H.1 Overall Significance Tests 9
H.2 Design Component Gain Analysis 10

I Image-Only Ablation **10**
I.1 Image-only Performance 10
I.2 Contribution of Chart Code 10

J Complexity Analysis Details **10**
J.1 Chart Complexity Indicators 10
J.2 Complexity Regression Analysis 10
J.3 Detailed Complexity Heatmaps 12

K Runtime Error and Chart Fidelity Violation Statistics **12**

L Representative Error Cases **13**
L.1 Rule-based Evaluation Cases 13
L.2 LLM-judged Evaluation Cases 13

M Human Evaluation and LLM-Judge Validation Details **14**
M.1 Human Alignment Across Model Capability Levels . 14
M.2 Human Alignment Across Instruction Levels 14
M.3 Cross-Judge Consistency and Judge-Model Bias . . . 14
M.4 Human Alignment Across Model Capability Levels . 15
M.5 Human Alignment Across Instruction Levels 15
M.6 Cross-Judge Consistency and Judge-Model Bias . . . 15

N Detailed Results for Cross-Representation Generalization **15**

O Use of AI Assistants **15**

P Gallery of Representative Error Cases **15**

A Comparison with Related Benchmarks

We compare ChartAnno with representative benchmarks from two related task families: chart editing tasks and annotation tasks.

A.1 Comparison with Chart Editing Tasks

Instruction and Code Complexity.. We further compare ChartAnno with ChartEdit, the closest code-based chart editing benchmark, in terms of instruction and target code complexity. As shown in Tab. 5, ChartAnno contains longer instructions and target code, suggesting additional complexity in annotation generation.

Table 5: Instruction and code length comparison.

Benchmark	Instr.	Code
ChartEdit	20.08	758.12
ChartAnno (Operation)	95.06	1535.68

Annotation Source Analysis.. Following prior studies on visualization annotation practices and design spaces [36, 37], we categorize annotation sources into three non-exclusive types: Internal (information available from the chart), Derived (information computed or inferred from data), and External (information obtained outside the chart).

Tab. 6 summarizes the distribution of annotation sources in ChartAnno.

Table 6: Distribution of annotation sources. Categories are non-exclusive.

Source	Charts	Ratio
Internal	595	49.58%
Derived	543	45.25%
External	918	76.50%

We further estimate the proportion of instruction instances requiring reasoning beyond direct visual modification. At least 63.5% of annotation instances involve non-trivial reasoning requirements:

$$\frac{1200 + 2 \times 543}{3600} = 63.5\%.$$

This conservative estimate considers two sources of reasoning. First, all Intent-level instructions require inferring communication goals and annotation strategies. Second, derived annotations in Operation-level and Implementation-level instructions require identifying relevant values or patterns from the underlying data before generating annotations. Therefore, a substantial portion of ChartAnno requires reasoning about why and how annotations should be introduced, beyond applying predefined visual edits.

A.2 Comparison with Annotation Tasks

We further examine AnnoBench [49], focusing on its 58 professional charts whose associated annotation tasks are derived from the original real-world visualizations. We reconstruct the corresponding annotated GT references through LLM-assisted conversion and manual refinement, and manually review the instructions, released code or data, and reconstructed GT references for all 58 samples.

We find that 30 samples (51.7%) cannot be directly evaluated under our GT-referenced Operation- and Implementation-level protocol for one or more of three reasons, although most remain compatible with AnnoBench's reference-free evaluation protocol. As these categories overlap, 28 samples remain for direct comparison under our protocol. The examples below illustrate these cases together with the corresponding annotation outputs generated by Claude Sonnet 4.6.

- *Ambiguous or Unspecified Instructions* ($n = 21$): annotation details such as components, text, color, position, or range are not specified. These samples remain compatible with reference-free Intent-level evaluation, but not with our GT-referenced Operation- and Implementation-level evaluation, as shown in Fig. 13.
- *Instruction–GT Mismatches* ($n = 7$): the instruction content does not directly correspond to the reconstructed GT or source chart, so these samples cannot be evaluated against the GT reference, as illustrated in Fig. 14.
- *Code/Data Issues* ($n = 4$): required information, such as labels referenced by the instruction, is absent from the released code or data, preventing faithful reproduction of the requested annotation, as shown in Fig. 15.



Figure 13: Example of limited specification in AnnoBench. The instruction omits the required annotation text, making it impossible to reproduce the original chart.

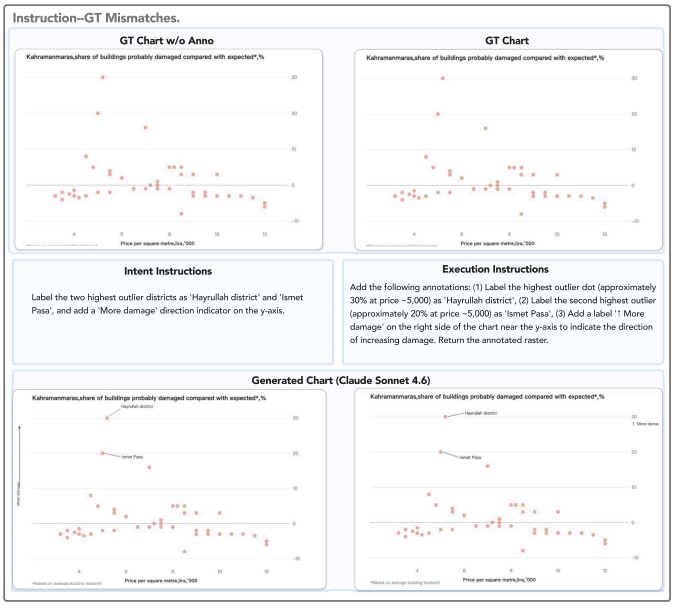


Figure 14: Example of an instruction–GT mismatch in AnnoBench. The instruction requires reproducing annotation elements that are absent from the original chart.

Complexity Comparison with ChartAnno.. We compare these 28 AnnoBench samples with the 120-chart ChartAnno subset from



Figure 15: Example of a code/data limitation in AnnoBench. The released code lacks the required labels, making it impossible to reproduce the original chart.

our cross-representation experiments. Compared with AnnoBench, CHARTANNO has 1.69× longer instructions (86.6 vs. 51.2 words) and 2.79× more annotation elements per chart (20.55 vs. 7.36). Its average code token increment is also 2.19× larger for D3 (1,374.2 vs. 627.1) and 3.87× larger for SVG (2,612.7 vs. 674.8). These differences indicate greater task complexity in CHARTANNO, particularly in annotation density and required code modification.

B DATA CONSTRUCTION DETAILS

This section provides implementation details for the data construction process described in Sec. 3.2 of the main paper.

B.1 Source Licenses and Intended Use

Existing chart sources are used as references for manual reconstruction and annotation revision, rather than directly redistributed as original images. Our use is limited to research and evaluation and is compatible with the public benchmark or scholarly use contexts of the source artifacts. The released artifact is intended for research and evaluation. Tab. 7 summarizes the licenses and access conditions of the source benchmarks and datasets.

Table 7: Licenses and access conditions of source benchmarks and datasets.

Source	License / Access Condition
ChartQAPro [29]	MIT license.
CharXiv [52]	Data CC BY-SA 4.0; code Apache 2.0.
ChartMimic [55]	Data and codebase Apache-2.0.
MatPlotBench [57]	Public benchmark; license not explicitly specified.
Rahman et al. [37]	Paper and supplemental materials under CC BY 4.0.
Stokes et al. [42]	Paper and supplemental materials under CC BY 4.0.
Recent arXiv papers	CC BY 4.0.

B.2 Manual Selection and Dataset Coverage

The review is conducted in multiple rounds. We then review candidates by chart type to reduce visual and structural redundancy. The final round verifies that the selected charts satisfy the above criteria.

Tab. 8 reports the distribution of chart types across data sources. CM, R, CQA, CX, MPB, and S denote ChartMimic, Rahman et al., ChartQAPro, CharXiv, MatPlotBench, and Stokes et al., respectively. Darker cells indicate more samples.

Table 8: Distribution of chart types across data sources. Darker cells indicate more samples.

Chart Type	CM	R	CQA	CX	MPB	S	ArXiv	Total
Area	0	13	15	0	0	4	1	33
Bar	41	19	114	6	0	12	21	213
Box	13	0	0	1	0	0	2	16
Combination	0	0	25	0	0	1	5	31
Contour	15	0	0	0	0	0	0	15
Density	19	0	0	0	0	0	0	19
Dot	0	0	4	0	0	2	0	6
Error point	46	0	0	0	0	1	0	47
Heatmap	28	0	1	6	0	0	9	44
Histogram	24	6	1	0	0	0	1	32
Line	25	17	144	9	1	17	34	247
Multi-plot	33	0	133	27	1	5	104	303
Pie	20	13	6	0	2	0	3	44
Radar	8	5	0	0	0	0	6	19
Scatter	21	15	14	4	0	10	30	94
Treemap	15	1	0	0	0	1	0	17
Violin	19	0	0	0	0	1	0	20
Total	**327**	**89**	**457**	**53**	**4**	**54**	**216**	**1200**

B.3 Reconstruction Quality and Correction Process

We provide additional details on the reconstruction quality evaluation and correction process described in Sec. 3.2 of the main paper. For charts from sources other than ChartMimic, we evaluate the initial reconstructions using GPT-5.4 [33] as the judge and a 100-point rubric adapted from ChartMimic [55]. The rubric assesses consistency with the source image across six dimensions: Chart Type, Layout, Text Content, Data, Style, and Clarity.

Tab. 9 reports the per-dimension average scores of the initial reconstructions.

Table 9: Average reconstruction quality scores for initial reconstructions of charts from sources other than ChartMimic. The total score is computed out of 100.

Dimension	Average Score
Total	92.34
Chart Type	19.99
Layout	9.88
Text Content	17.78
Data	17.95
Style	16.91
Clarity	9.83

Following this evaluation, we manually refine the reconstructed charts to improve their consistency with the corresponding source images. We additionally inspect the source annotations and correct three types of issues that could compromise the quality of the resulting references.

Ambiguous Annotation Intent. Some annotations have targets, directions, or connections that are unclear or inconsistent with the surrounding chart content. We inspect the chart content and underlying data to recover the intended relation, and revise the annotation text, anchor position, direction, or connection accordingly. Fig. 16 shows representative examples.

Factual Inconsistencies. Some annotation text conflicts with the underlying data or derived statistics, such as incorrect percentage changes



Figure 16: Examples of correcting ambiguous annotation intent. The left side shows annotations with unclear or inconsistent targets, directions, or connections; the right side shows the corrected versions.

or summary values; we recompute the relevant quantities and revise the text accordingly, as shown in Fig. 17.

Visual Presentation Issues. Some annotations are difficult to interpret because of presentation problems such as truncated text, occluded labels, or unclear placement. When the intended meaning is recoverable, we correct these issues by adjusting the annotation position, style, or text. If an annotation cannot be corrected without introducing ambiguity, we remove it from the reference chart. Fig. 18 shows representative examples.

C Rule-based Evaluation Details

C.1 Execution Environment

Generated programs are executed in a sandboxed Python environment with Python 3.12.13, Matplotlib 3.10.8, NumPy 2.2.6, and Pandas 3.0.2. Each program is executed with a timeout of 120 seconds. Rendered charts are saved as JPG images using the output settings specified by each generated program. Programs that fail to execute or exceed the timeout receive zero for all downstream quality metrics.

C.2 Rendered Element Representation

The rule-based metrics build on the differential annotation extraction and classification procedure described in Sec. 3.3.1 of the main paper, which matches elements in the annotated and generated charts against the unannotated counterpart and classifies unmatched elements into seven annotation categories. The following subsections present the corresponding implementations: annotation element extraction, chart fidelity, annotation matching, and color matching.

C.3 Annotation Element Extraction

We summarizes the annotation extraction process. The extractor identifies elements from rendered charts, removes elements already present in the unannotated chart, and groups the remaining elements into seven annotation categories.

Listing 1: Annotation Element Extraction

```
def extract_annotations(annotated_fig, unannotated_fig):
    base_elements = extract_visible_elements(
     unannotated_fig)
```



Figure 17: Examples of correcting factual inconsistencies. The left side shows annotation text that conflicts with the underlying data or derived statistics; the right side shows the corrected versions.

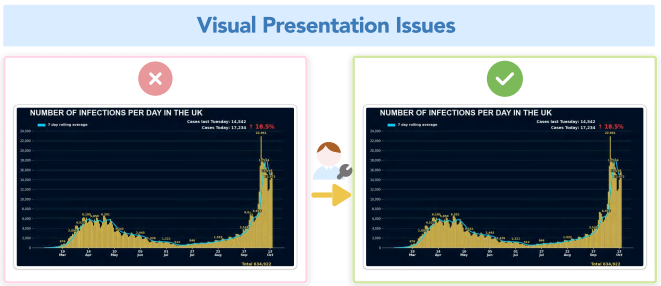


Figure 18: Examples of correcting visual presentation issues. The left side shows truncated, occluded, or otherwise unclear annotations; the right side shows the corrected versions.

```
    anno_elements = extract_visible_elements(annotated_fig)

    added_elements = diff_elements(anno_elements,
     base_elements)

    annotations = {
        "enclosure": [],
        "connector": [],
        "text": [],
        "glyph": [],
        "color": [],
        "indicator": [],
        "geometric": [],
    }

    for element in added_elements:
        if is_enclosure(element):
            annotations["enclosure"].append(record(element)
     )
        elif is_connector(element):
            annotations["connector"].append(record(element)
     )
        elif is_text(element):
            annotations["text"].append(record(element))
        elif is_glyph(element):
            annotations["glyph"].append(record(element))
        elif is_color(element):
            annotations["color"].append(record(element))
        elif is_indicator(element):
            annotations["indicator"].append(record(element)
     )
        elif is_geometric(element):
            annotations["geometric"].append(record(element)
     )

    return annotations
```

We identify the seven annotation categories using rules based on element type, visibility, geometry, color, and position relative to chart regions or text.

Enclosure. Enclosure refers to visual regions that surround, group, or emphasize chart content. We extract highlighted regions, background areas, framed regions, and other visual boundaries. To avoid confusing data marks with annotations, we exclude ordinary data representations and retain elements that provide additional emphasis beyond the original chart encoding.

Connector. Connector captures visual links between annotations and their targets. We extract arrows, leader lines, curved connectors, and guide lines associated with annotation elements. Long lines spanning the full chart range are not treated as connectors because they usually represent reference structures rather than annotation links.

Text. Text includes newly added semantic annotation text. We extract visible text elements with non-empty content and record their text content, bounding regions, chart regions, and color. For text annotations with attached connectors, we separate the text region from the connecting component to avoid counting them as a single element. Text elements already present in the unannotated chart, such as axis labels and legends, are excluded by comparison with the unannotated baseline.

Glyph. Glyph captures local symbols that mark specific data points or regions. Examples include added markers, highlighted points, symbols, and emphasis marks. The extractor avoids treating dense data marks as annotations and retains only sparse or visually distinctive added elements.

Color. Color captures sparse color changes that carry annotation meaning. We do not record every color used in the chart. Instead, we identify colors that are newly introduced or selectively applied compared with the original chart. Default colors, background colors, and ordinary data-series colors are filtered out when they do not function as annotation highlights.

Indicator. Indicator refers to auxiliary structures that mark values, ranges, thresholds, or statistical references. We extract reference lines, threshold markers, baselines, bracket-like structures, and other visual references. Unlike connectors, indicators typically do not link a text label to a specific target but instead highlight values, intervals, or structural properties.

Geometric. Geometric annotation captures structural components that change the visual organization of the chart. This includes inset regions, zoom-related structures, and exploded chart components. These elements are treated separately because they are not well represented by simple text, line, or region categories.

C.4 Chart Fidelity

Listing 2: Chart Fidelity

```
def chart_fidelity(unannotated_fig, predicted_fig):
    base_spec = extract_protected_chart_spec(
     unannotated_fig)
    pred_spec = extract_protected_chart_spec(predicted_fig)

    if not match_figure_ratio(base_spec, pred_spec):
        return 0

    if not match_chart_layout(base_spec, pred_spec):
        return 0

    if not preserve_data_marks(base_spec, pred_spec):
        return 0

    return 1
```

C.5 Annotation Matching

Listing 3: Annotation Matching

```
def annotation_matching(gt_annotations, pred_annotations):
    total_intersection = 0
    total_union = 0

    for category in ANNOTATION_CATEGORIES:
        gt_items = gt_annotations[category]
        pred_items = pred_annotations[category]

        if category == "text":
            intersection, union = match_text_items(gt_items,
      pred_items)
        else:
            intersection = min(len(gt_items), len(
     pred_items))
            union = max(len(gt_items), len(pred_items))

        total_intersection += intersection
        total_union += union

    return safe_divide(total_intersection, total_union,
     empty_value=1.0)
```

C.6 Color Matching

Listing 4: Color Matching

```
def color_matching(gt_annotations, pred_annotations):
    gt_colors = extract_colors_by_category(gt_annotations)
    pred_colors = extract_colors_by_category(
     pred_annotations)

    total_similarity = 0.0
    total_gt = 0
    total_pred = 0
```

```
    for category in ANNOTATION_CATEGORIES:
        gt_group = gt_colors[category]
        pred_group = pred_colors[category]

        pairs = maximum_bipartite_matching(
            gt_group,
            pred_group,
            score_fn=color_similarity,
        )

        total_similarity += sum_pair_scores(pairs)
        total_gt += len(gt_group)
        total_pred += len(pred_group)

    precision = safe_divide(total_similarity, total_pred,
     empty_value=1.0)
    recall = safe_divide(total_similarity, total_gt,
     empty_value=1.0)
    return f1_score(precision, recall)
```

C.7 Text Relationship Statistics

As described in Sec. 3.3 of the main paper, text-overlap and off-canvas statistics are provided to the LLM judge as quantitative evidence for visual clarity rather than as standalone scores. For each newly added text element, we compute its maximum overlap ratio with other visible text elements. The overlap ratio is measured relative to the area of the added text element, reflecting how much of the annotation is visually obstructed. We also compute the off-canvas ratio of each added text element, measuring the portion of its bounding region that falls outside the figure canvas.

We further aggregate these quantities into global overlap and off-canvas statistics. The global overlap statistic measures the total area affected by text conflicts relative to the figure canvas. The global off-canvas statistic measures the total area of added text that falls outside the canvas. These statistics help the judge identify text crowding, severe overlap, and off-canvas annotations, while the final score remains based on the full rendered chart.

D LLM-Judged Evaluation Rubric

This appendix provides the detailed scoring rubric used by the LLM-based judge (Sec. 3.3.2 of the main paper). Beyond the integer scale from 1 to 5, with 0 for failed or missing cases, a score of 1 indicates a severe failure; 2 indicates clear problems below the baseline acceptable level; 3 indicates baseline acceptability; 4 indicates clearly above-baseline quality; and 5 indicates the highest quality. The judge is instructed to score strictly, evaluate each metric independently, and choose the lower score when uncertain.

D.1 Semantic Faithfulness

Table 10: Scoring rubric for Semantic Faithfulness.

Score	Criterion
0	Required annotation is missing or cannot be judged.
1	Severe semantic failure, including wrong target, incorrect trend/value/relation/conclusion, missing required text, factual deviation, misreference, or over-generation.
2	Noticeable semantic problems; the main meaning is partly preserved but contains local errors, incomplete execution, inaccurate expression, or partial omission.
3	Baseline acceptable; target, text, trend, value, relation, and conclusion are basically correct, with only minor imperfections.
4	Above baseline; key targets, text, relations, scope, and structure are correct, with no substantive omission, misreference, or factual deviation.
5	Fully satisfies all semantic requirements accurately and rigorously, consistent with the instruction and reference when applicable.

D.2 Semantic Clarity

Table 11: Scoring rubric for Semantic Clarity.

Score	Criterion
0	Required annotation is missing or cannot be judged.
1	Severe clarity failure, including unclear referents, wrong or unstable text-object relations, misleading encoding, or conflicting interpretations.
2	Noticeable ambiguity in target, relation, or meaning, requiring extra inference, pause, or confirmation from the reader.
3	Baseline acceptable; referents are basically clear, the text-object relation is followable, and only minor local ambiguity remains.
4	Above baseline; object, relation, and meaning are clear, stable, and non-misleading, requiring almost no extra confirmation.
5	Meaning is immediately clear, with a natural and unambiguous relation between annotation text and visual objects.

D.3 Visual Clarity

Table 12: Scoring rubric for Visual Clarity.

Score	Criterion
0	Required annotation is missing or cannot be judged.
1	Severe clarity failure, including overlap with essential chart text, occlusion of key data content, severe overlap, off-canvas or clipped text above 10%, or moderate overlap above 50%.
2	Noticeable clarity problems, including crowding, moderate overlap, partial blocking, non-severe clipping/off-canvas issues, 20%–50% moderate overlap, or `anno_overlap_pct` $\geq$ 40%.
3	Baseline acceptable; the chart is basically readable, with only minor local crowding or readable overlap in non-critical areas.
4	Above baseline; annotations and chart content are clearly separated, with no visible overlap, clipping, occlusion, or intrusion into important information.
5	Reading is smooth, with almost no perceptible clarity issue and no intrusion into important chart information.

D.4 Annotation Organization Quality

Table 13: Scoring rubric for Annotation Organization Quality.

Score	Criterion
0	Required annotation is missing or cannot be judged.
1	Severe organization failure, including chaotic placement, wrong grouping, severe target detachment, out-of-body annotations, colors that harm recognition, or more than 50% of annotation groups containing four or more elements.
2	Noticeable organization problems, including loose placement, weak target attachment, redundancy, overlap, imbalance, poor spacing, weak contrast, confusing color use, or more than 20% of annotation groups containing four or more elements.
3	Baseline acceptable; layout is generally reasonable, annotations are integrated with the chart body, and colors are visible and acceptable.
4	Above baseline; grouping, target attachment, spacing, hierarchy, and colors are well coordinated, with no obvious redundancy or over-complexity.
5	Excellent organization; composition is balanced, natural, easy to parse, and supported by effective color use.

D.5 Attention Guidance

Table 14: Scoring rubric for Attention Guidance.

Score	Criterion
0	Required annotation is missing or cannot be judged.
1	Severe guidance failure, including attention drawn to the wrong region, buried main target, or strongly misleading visual focus.
2	Noticeable guidance problems, including weak salience, competing highlights, hierarchy confusion, unstable focus, or substantial search effort.
3	Baseline acceptable; the main target can be found and the guidance basically works, though some browsing or brief pause may be needed.
4	Above baseline; the main target is salient, hierarchy is clear and stable, and the focus can usually be grasped quickly.
5	The main target is extremely clear, with efficient and natural guidance that can be identified almost immediately.

E LLM-Judged Evaluation Prompt

We use the same evaluation prompt template across all models and input settings to score *Semantic Consistency* and *Design Effectiveness*. The scoring rubric is fixed. Only the placeholder `{stage_constraints}` is replaced according to the three instruction levels: Intent, Operation, and Implementation.

E.1 Evaluation Prompt Template

Prompt for Semantic and Design Evaluation

Role. You are an expert judge for chart annotation quality.

Task. Evaluate only the added annotations in the AI-generated chart, not the base chart design. Use the instruction, AI image, Text_Relation_Results, and GT image to judge whether the annotations are semantically correct, visually clear, well organized, and effective in guiding attention. Use the GT image according to the instruction-level constraints specified below. Text_Relation_Results provides text-overlap and off-canvas statistics as supporting evidence.

Scoring Policy.

- Use a 0–5 rating scale for each metric: 0 = completely missing or extremely poor; 1 = very poor; 2 = below the acceptable level; 3 = acceptable; 4 = clearly above acceptable; 5 = highest quality.
- Score strictly. If uncertain, choose the lower score.
- Score each metric independently.
- Return JSON only.

Semantic Consistency. Judge the following two metrics based on the instruction and the generated annotations, using the GT image as a reference when needed.

1. Semantic Faithfulness (sf)
Description: Evaluate whether the annotations follow the intended meaning of the instruction.
Check: target object, specified text, trend, value, relation, conclusion, missing required content, wrong reference, factual error, over-generation, wrong visual encoding, and misleading grouping.
Level:

- **1:** Clear semantic failure, such as wrong target, wrong trend/value/relation/-conclusion, missing key content, major factual error, wrong reference, over-generation, or wrong encoding.
- **2:** Noticeable semantic problems. The main meaning is partly preserved, but local errors, incomplete execution, inaccurate value/trend/relation/text, or partial omission remain.
- **3:** Acceptable. The target object and main meaning are basically correct, with only minor non-critical problems.
- **4:** Clearly above acceptable. Key targets, text, meaning, relations, scope, and structure are correct, with at most trivial problems.
- **5:** All semantic requirements of the instruction are fully and accurately satisfied, with correct targets, content, relations, scope, and visual encoding.

2. Semantic Clarity (sc)
Description: Evaluate whether the relation between annotation text and visual objects is clear.
Check: clear referent, text-object relation, ambiguity, competing interpretations, and need for extra inference.
Level:

- **1:** Clear failure, including unclear referents, wrong or unstable text-object relations, obvious ambiguity, misleading encoding, or competing interpretations.
- **2:** Noticeable problems, including locally unclear object, relation, or meaning that requires extra inference, pause, or confirmation.
- **3:** Acceptable. Referents are clear and the text-object relation is basically followable, with only minor ambiguity.
- **4:** Clearly above acceptable. Object, relation, and meaning are clear, stable, and not misleading.
- **5:** Meaning is immediately clear; the relation between text and visual objects is natural and easy to understand.

Design Effectiveness.

3. Visual Clarity (vc)
Description: Evaluate whether annotations interfere with reading the chart. Focus on overlap, clipping, crowding, and blocking between annotations and chart content. Text_Relation_Results provides evidence, but direct visual judgment from the image has priority when visible blocking is present.
Check: overlap or clipping among annotations; blocking between annotations and chart content; blocked titles, subtitles, axis labels, legends, tick labels, data marks, or key labels; and readability of key chart information.
Level:

- **1:** Severe clarity failure, including severe overlap, intrusion into essential chart text, visible blocking of key chart content, obvious clipped/off-canvas text, or unreadable placement. Typical triggers include any severe overlap that makes text unreadable, off-canvas or clipped text $> 10\%$, or moderate overlap $> 50\%$.
- **2:** Noticeable clarity problems, including visible crowding, conflict, moderate overlap, partial blocking, competition with chart content, partial blocking of non-critical data marks, or non-severe clipping/off-canvas issues. Typical triggers include 20%–50% moderate overlap or `anno_overlap_pct` $\geq 40\%$.
- **3:** Acceptable. The chart is basically readable, with only minor local crowding or mild readable overlap in non-critical areas; key chart text and data content remain unobstructed.
- **4:** Clearly above acceptable. Annotations and chart content are clearly separated, with no visible overlap, clipping, blocking, or intrusion into important chart information.
- **5:** Reading is smooth, with almost no visible clarity issue and no intrusion into important chart information.

4. Annotation Organization Quality (org)
Description: Evaluate whether annotations are well placed and visually fit the chart. Focus on placement, grouping, spacing, hierarchy, target attachment, and color use. This metric penalizes poor organization even when the chart remains readable.
Check: placement, grouping, hierarchy, spacing, whitespace, target attachment, redundancy, disorder, overly complex annotation groups, color contrast, separability, coordination, and support for emphasis.
Level:

- **1:** Clear organization failure, including chaotic placement, clearly wrong grouping, annotations largely outside the chart body, severe detachment from targets, or colors that seriously harm recognition or create annotation-data confusion. This includes cases where more than 50% of annotation groups contain four or more elements.
- **2:** Clear organization or color problems, including loose or scattered placement, intrusion into title/subtitle/axis/legend areas, placement outside the main plot area or chart body, weak target attachment, redundancy, local imbalance, poor spacing, weak contrast, poor separability, or confusing color use. This includes cases where more than 20% of annotation groups contain four or more elements.
- **3:** Acceptable. The layout is generally reasonable; annotations are within or visually integrated with the main chart body, close enough to targets, readable, and acceptable in color use.
- **4:** Clearly above acceptable. Grouping, target attachment, spacing, hierarchy, and colors are well coordinated, with no intrusion into essential chart-text areas.
- **5:** Excellent organization. The composition is balanced, natural, and easy to parse, with effective color use and no intrusion into essential chart-text areas.

5. Attention Guidance (attn)
Description: Evaluate whether the annotations highlight the intended target or target set and create a clear visual focus.

Check: target salience, hierarchy clarity and stability, competing highlights or misleading focus, and search effort needed to find the focus.
Level:

- **1:** Clear failure, including attention drawn to the wrong place, main target severely buried, or strongly misleading focus.
- **2:** Noticeable problems, including weak salience, competing highlights, hierarchy confusion, unstable focus, or substantial search effort.
- **3:** Acceptable. The main target can be found and the guidance basically works, but salience and hierarchy are ordinary; brief search may be needed.
- **4:** Clearly above acceptable. The main target is fairly salient, hierarchy is clear and stable, and the focus can usually be grasped quickly.
- **5:** The main target is extremely clear; visual guidance is efficient and natural, and the focus can be identified almost immediately.

Constraints.
`{stage_constraints}`

Output Format. Return exactly one JSON object using the following schema, where each metric value is an integer score from 0 to 5. Do not output any extra text.

```
{
  "results": [
    {
      "sf": 0,
      "sc": 0,
      "vc": 0,
      "org": 0,
      "attn": 0
    }
  ]
}
```

Inputs.

- Instruction: `{instruction}`
- Text_Relation_Results: `{text_relation_results}`
- AI image: the chart image generated by the evaluated model
- GT image: the reference annotated chart image

E.2 Instruction-Level Constraint Instantiations

The placeholder `{stage_constraints}` is instantiated according to the instruction level. All metric scores must be integers in $[0, 5]$. Do not evaluate visual similarity to the GT image. Alternative annotation types, layouts, colors, and placements should not be penalized solely for differing from the GT image, provided that they faithfully satisfy the instruction and are semantically clear and visually effective. For Operation- and Implementation-level instructions, the GT image is used as a comparison reference, and a substantial mismatch with the GT image caps the corresponding metric at 3.

F Generation Prompts and Examples

F.1 Generation Prompt Templates

We build each prompt from five parts: a mode-specific role description, common constraints, the unannotated chart code, an instruction-level description, and the annotation instruction. The common constraints and final request are fixed across all models, instruction levels, and input settings.

Mode-specific role descriptions. We use different role descriptions for the two input settings.

Code Input Role

You are an expert in chart annotation and Python visualization. I have created a figure but have not added any annotations yet. I will provide you with the chart code without annotations, along with annotation instructions. Your task is to modify the provided code to add annotations based on the instructions.

Code + Image Input Role

You are an expert in chart annotation and Python visualization. I have created a figure but have not added any annotations yet. I will provide you with the chart code without annotations, along with its corresponding reference image and annotation instructions. The image is generated by the code and is provided to help with positioning and alignment. Your task is to modify the provided code to add annotations based on the instructions, using the image for guidance.

Common constraints and code input. After the role description, all prompts include the same constraints. These constraints require executable, self-contained Python code and discourage unnecessary changes to the base chart. For multi-plot figures, we also specify the subplot indexing order. The unannotated chart code is then inserted into the shared template.

Common Constraints

You must follow these constraints:

1. The code must be self-contained and executable.
2. Use Python, and do not change the style of the base chart.
3. Output code only. Do not include comments or explanations.

Multi-plot Constraint

Layout Constraint for Multi-plots: Subplots are indexed from 0, ordered left to right and top to bottom in row-major order (0, 1, 2...).

Shared Code Block

Here is the Python code without annotations:
`{code}`

Instruction-level descriptions. We add a short description before each annotation instruction to specify the expected level of detail.

Intent-level Description

I will describe the communication goal and the key information to be conveyed, without specifying how the visualization should look. You should interpret this intent, choose an appropriate annotation design, and generate Matplotlib code accordingly.

Operation-level Description

I will describe the annotation actions and spatial relations using qualitative instructions, such as relative positions, layout relations, and non-numeric style descriptions. You should convert these instructions into Matplotlib code.

Implementation-level Description

I will provide concrete implementation details, including coordinates, colors, sizes, and other rendering parameters. You should follow these specifications precisely and produce Matplotlib code that implements them.

Instruction and final request. Finally, we insert the instance-specific annotation instruction and ask the model to return complete Python Matplotlib code.

Instruction and Final Request

`{level_description}`
Instruction:
`{instruction}`
Based on my instruction above, please write the complete Python Matplotlib code for me.

F.2 Prompt and Evaluation Example

Fig. 19 shows one full generation and evaluation example using Gemini 3.1 Pro Preview. The example includes the two input settings, the shared prompt structure, the three instruction levels, generated charts, the ground-truth annotated chart, and the corresponding rule-based and LLM-judged results. It also shows how the instruction becomes more specific from Intent to Operation and Implementation.

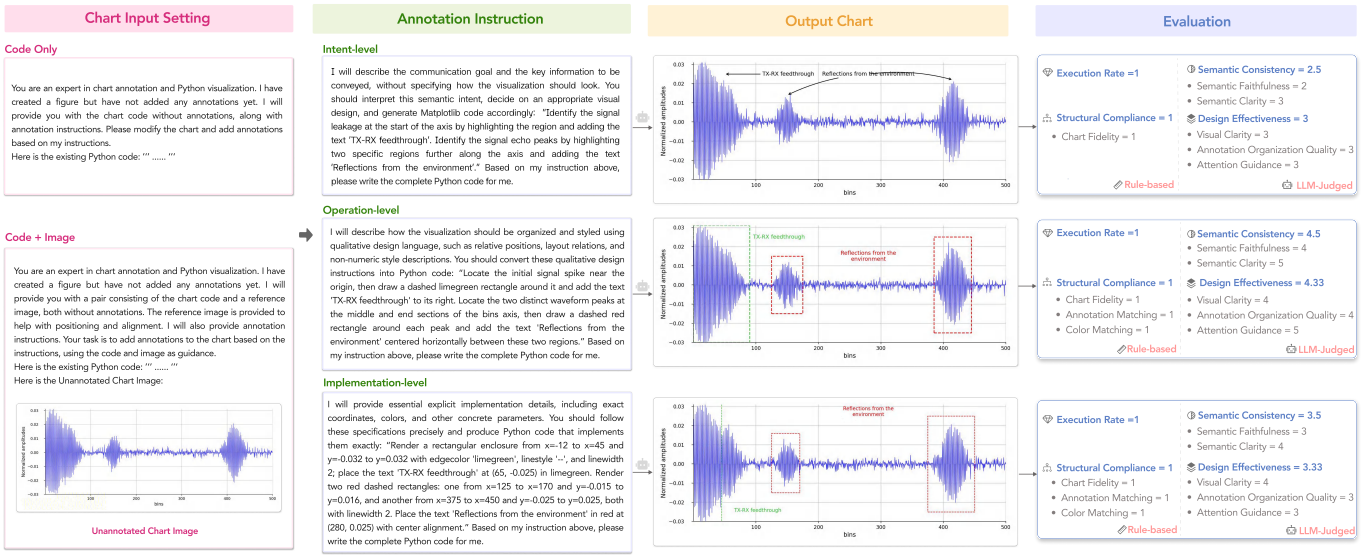


Figure 19: Prompt and evaluation example for one chart instance using Gemini 3.1 Pro Preview.

G Model Configurations and Licenses

G.1 Model Versions and Inference Settings

Tables 15 and 16 summarize the model versions and decoding settings. Proprietary models are identified by their release versions, and open-source models are pinned to the Hugging Face checkpoints listed in Tab. 15, which are used throughout all evaluations. Each row of Tab. 16 corresponds to the same model as Tab. 15, and entries marked with "–" indicate settings that are unavailable or left at their defaults as described below. The maximum output length is set to 16,384 tokens, except for Gemini models, for which we use the default output-length setting to avoid API-side truncation behavior.

For API-based models, `do_sample` is not exposed; for locally deployed models, we set `do_sample=False`. Claude Sonnet 4.6 does not support setting temperature and top-p simultaneously, so only temperature is specified. We do not explicitly control reasoning-related settings. Gemini models enable reasoning by default, while GPT-5.4 and Claude Sonnet 4.6 are evaluated under their default reasoning-related settings.

Table 15: Model versions and Hugging Face checkpoints.

Model	Version / HF Checkpoint
Proprietary Models	
GPT-5.4	gpt-5.4
Gemini 3.1 Pro Preview	gemini-3.1-pro-preview
Gemini 3 Flash Preview	gemini-3-flash-preview
Claude Sonnet 4.6	claude-sonnet-4-6
Open-Source Models	
Kimi K2.5	moonshotai/Kimi-K2.5
Gemma 4 31B	google/gemma-4-31B
Qwen3.5-397B-A17B	Qwen/Qwen3.5-397B-A17B
Qwen3.5-122B-A10B	Qwen/Qwen3.5-122B-A10B
Qwen3.5-27B	Qwen/Qwen3.5-27B
Qwen3.5-9B	Qwen/Qwen3.5-9B

Table 16: Inference settings for all models. Unset values indicate default or unavailable settings.

Model	Do Sample	Max Tokens	Temp.	Top-p
Proprietary Models				
GPT-5.4	–	16384	0	1
Gemini 3.1 Pro Preview	–	Default	0	1
Gemini 3 Flash Preview	–	Default	0	1
Claude Sonnet 4.6	–	16384	0	–
Open-Source Models				
Kimi K2.5	–	16384	–	–
Gemma 4 31B	False	16384	0	1
Qwen3.5-397B-A17B	False	16384	0	1
Qwen3.5-122B-A10B	False	16384	0	1
Qwen3.5-27B	False	16384	0	1
Qwen3.5-9B	False	16384	0	1

G.2 Computational Infrastructure and Budget

The local evaluation uses approximately 450 GPU hours.

G.3 Model Licenses

Proprietary models and Kimi K2.5 are accessed through their official APIs and used under the corresponding API terms, while all locally deployed open-source models (Gemma 4 31B and the Qwen3.5 series) are released under the Apache 2.0 license for both model weights and code.

H Chart Image Input Analysis

This section provides additional evidence for the effect of chart image input. We first report overall significance tests comparing Code + Image Input with Code Input, and then break down the normalized gain in *Design Effectiveness* into its three components.

H.1 Overall Significance Tests

We test whether Code + Image Input improves over Code Input using all matched instances across the ten models. For *Execution Rate*, we use a one-sided exact McNemar test. For *Structural Compliance*, *Semantic Consistency*, and *Design Effectiveness*, we use one-sided paired t-tests.

Tab. 17 reports the overall results.

Table 17: Overall score changes from Code Input to Code + Image Input.

Metric	Code	Code+Image	Δ	p	Effect Size
Execution Rate	0.941	0.950	0.009	$< .001$	$g = 0.080$
Structural Compliance	0.817	0.822	0.005	$< .001$	$d_z = 0.022$
Semantic Consistency	3.369	3.374	0.005	.209	$d_z = 0.004$
Design Effectiveness	3.418	3.439	0.021	$< .001$	$d_z = 0.021$

In Tab. 17, Δ denotes the mean score change from Code Input to Code + Image Input. *, **, and *** indicate $p < 0.05$, $p < 0.01$, and $p < 0.001$, respectively.

H.2 Design Component Gain Analysis

Fig. 20 breaks down the normalized gains from chart image input into the three components of *Design Effectiveness*. *Visual Clarity* shows the largest and most consistent gains across models, while *Annotation Organization Quality* also improves in many cases. In contrast, *Attention Guidance* shows limited or negative gains for most models, indicating that chart images help more with local layout and readability than with guiding attention to the intended insight. The gains are also model-dependent: GPT-5.4 shows negative gains across all three components, whereas several open-source models, especially Qwen variants, benefit more from image input, mainly in *Visual Clarity*.

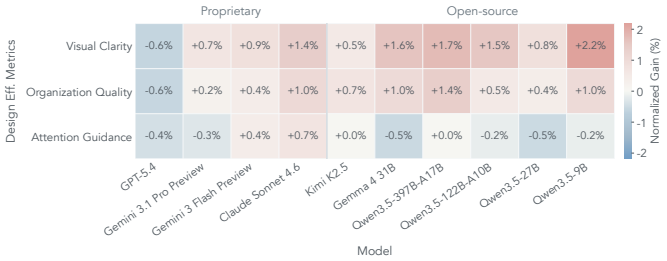


Figure 20: Normalized gains from chart image input on *Design Effectiveness* components. Positive values indicate improvements over Code Input.

I Image-Only Ablation

I.1 Image-only Performance

Tab. 18 reports the results under the Image-only setting across three instruction levels.

To further investigate the contribution of different input modalities, we evaluate an Image-only setting under the same experimental protocol. We evaluate all 10 models on the full dataset.

Compared with the Code Input setting in Tab. 3 of the main paper, Image-only performance is substantially lower, especially on semantic and design metrics. Although the relative ranking remains broadly similar, generating annotations from images alone requires additional capabilities beyond annotation generation, including visual chart understanding, recovery of underlying data and structural information, and reconstruction of the original chart.

Among proprietary models, Gemini 3 Flash Preview achieves competitive performance on several Operation- and Implementation-level metrics, while Kimi K2.5 consistently performs best among open-source models.

I.2 Contribution of Chart Code

Tab. 19 reports the absolute improvements introduced by adding chart code.

J Complexity Analysis Details

J.1 Chart Complexity Indicators

We define six indicators to analyze how model performance changes with chart and annotation complexity. The main paper (Sec. 4.2.2) overviews these indicators and splits charts into simple, medium, and complex groups for each; here we detail how each indicator is computed. These indicators are used only for the complexity analysis and are not part of the main evaluation metrics.

Code Token Increment. Code Token Increment measures the extra code needed to add annotations. Following the tokenization procedure in Sec. 3.2 of the main paper, we compute code length with the Llama2 tokenizer and measure the difference between annotated and unannotated chart code. A larger value means that the annotations require more code changes.

Instruction Length. Instruction Length measures the average word count of the three instruction levels for each chart. A larger value means that the chart is paired with longer and more detailed annotation instructions.

Annotation Type Count. Annotation Type Count measures the number of distinct annotation element types in a chart. It is computed from the structured annotation elements described in Sec. C.3. This indicator reflects the variety of annotation forms rather than the total number of annotation elements.

Annotation Count. Annotation Count measures the total number of added annotation elements. It is also computed from the structured annotation elements described in Sec. C.3. Unlike Annotation Type Count, this indicator captures annotation quantity rather than type variety.

Annotation Spatial Distribution Entropy. Annotation Spatial Distribution Entropy measures how widely annotations are spread over the chart canvas. We compute it from the pixel-level difference between the annotated chart image and the unannotated chart image. We first mark pixels whose RGB difference exceeds a fixed threshold as annotation pixels. We then divide the canvas into a 4×4 grid and count annotation pixels in each cell. Let p_i denote the proportion of annotation pixels in the i-th grid cell. The normalized entropy is:

$$H_{\text{anno}} = \frac{-\sum_{i=1}^{K} p_i \log p_i}{\log K},$$

where $K = 16$. A higher value means that annotations are spread across more regions of the chart.

Visual Element Occupancy. Visual Element Occupancy measures the proportion of the chart canvas occupied by non-background visual content. We compute it from the unannotated chart image. For each image, we estimate the background color using the median RGB value of border pixels. Pixels close to the estimated background color are treated as background pixels. We compute the whitespace ratio R_{white} and define:

$$R_{\text{occ}} = 1 - R_{\text{white}}.$$

A larger value indicates that chart content occupies more of the canvas, leaving less empty space for annotations.

J.2 Complexity Regression Analysis

The main paper reports relative score drops from simple to complex cases. Here, we further run regression analysis to test whether these trends remain after controlling for chart category, input setting, and instruction level.

For each model, metric, and complexity indicator, we fit a categorical regression with the simple group as the reference:

$$y = \beta_0 + \beta_1 \mathbb{I}(\text{medium}) + \beta_2 \mathbb{I}(\text{complex}) + \gamma^{\top} Z + \epsilon,$$

where y is the metric score, and Z includes chart category, input setting, and instruction level. We report β_2 as the complex-vs-simple effect. A negative β_2 means that complex cases receive lower scores than simple cases after controlling for these factors. Significance levels are denoted by $^{*}p < 0.05$, $^{**}p < 0.01$, and $^{***}p < 0.001$.

We run separate regressions for Execution Rate, Structural Compliance, Semantic Consistency, and Design Effectiveness. The coefficients should be interpreted within each metric, because the metrics have different scales. Tables 20–25 report the complex-vs-simple coefficients for each complexity indicator.

Table 18: Results under the Image-only Input setting across three instruction levels. Exec., Struct., Sem., and Design denote *Execution Rate*, *Structural Compliance*, *Semantic Consistency*, and *Design Effectiveness*. Gray Struct.* columns report only Chart Fidelity for Intent-level instructions and are not directly comparable with *Structural Compliance* at the other levels.

Model	Intent-level				Operation-level				Implementation-level			
	Exec.	Struct.*	Sem.	Design	Exec.	Struct.	Sem.	Design	Exec.	Struct.	Sem.	Design
Proprietary Models												
GPT-5.4	0.976	0.140	3.000	3.127	0.965	0.360	2.796	3.175	0.952	0.395	2.938	3.307
Gemini 3.1 Pro Preview	0.947	0.137	3.080	3.239	0.941	0.375	2.874	3.273	0.938	0.409	2.958	3.373
Gemini 3 Flash Preview	0.953	0.113	3.062	3.203	0.957	0.372	2.882	3.292	0.955	0.410	3.070	3.469
Claude Sonnet 4.6	0.957	0.053	2.897	3.149	0.945	0.308	2.604	3.131	0.950	0.349	2.683	3.211
Open-Source Models												
Kimi K2.5	0.942	0.104	2.695	3.028	0.936	0.337	2.433	3.048	0.928	0.366	2.548	3.114
Gemma 4 31B	0.860	0.041	2.373	2.631	0.857	0.292	2.165	2.658	0.838	0.309	2.188	2.663
Qwen3.5-397B-A17B	0.821	0.053	2.220	2.415	0.807	0.269	1.970	2.399	0.820	0.300	2.019	2.474
Qwen3.5-122B-A10B	0.781	0.052	1.998	2.230	0.764	0.249	1.738	2.151	0.764	0.279	1.840	2.261
Qwen3.5-27B	0.700	0.055	1.813	2.017	0.676	0.211	1.593	1.984	0.677	0.224	1.660	2.032
Qwen3.5-9B	0.646	0.018	1.393	1.663	0.578	0.167	1.164	1.501	0.611	0.197	1.287	1.706

Table 19: Absolute improvements of Code + Image over Image-only. Positive values indicate additional benefits from providing chart code beyond rendered chart images. Gray Struct.* columns report Chart Fidelity gains only for Intent-level instructions.

Model	Intent-level				Operation-level				Implementation-level			
	Exec.	Struct.*	Sem.	Design	Exec.	Struct.	Sem.	Design	Exec.	Struct.	Sem.	Design
Proprietary Models												
GPT-5.4	+0.011	+0.716	+0.553	+0.294	+0.023	+0.481	+0.807	+0.522	+0.041	+0.502	+1.119	+0.830
Gemini 3.1 Pro Preview	+0.048	+0.780	+0.600	+0.400	+0.052	+0.501	+0.906	+0.637	+0.060	+0.507	+1.191	+0.857
Gemini 3 Flash Preview	+0.044	+0.583	+0.538	+0.393	+0.031	+0.443	+0.761	+0.524	+0.037	+0.476	+0.936	+0.675
Claude Sonnet 4.6	+0.027	+0.833	+0.615	+0.272	+0.029	+0.535	+0.894	+0.515	+0.038	+0.557	+1.337	+0.901
Proprietary Average	+0.033	+0.728	+0.576	+0.340	+0.034	+0.490	+0.842	+0.549	+0.044	+0.510	+1.146	+0.816
Open-Source Models												
Kimi K2.5	+0.041	+0.773	+0.615	+0.252	+0.029	+0.481	+0.900	+0.407	+0.044	+0.512	+1.330	+0.847
Gemma 4 31B	+0.099	+0.746	+0.786	+0.476	+0.083	+0.504	+1.055	+0.676	+0.100	+0.532	+1.493	+1.115
Qwen3.5-397B-A17B	+0.132	+0.780	+0.815	+0.574	+0.138	+0.512	+1.081	+0.802	+0.125	+0.538	+1.637	+1.296
Qwen3.5-122B-A10B	+0.148	+0.756	+0.846	+0.593	+0.154	+0.519	+1.147	+0.861	+0.157	+0.540	+1.618	+1.338
Qwen3.5-27B	+0.239	+0.773	+1.147	+0.883	+0.218	+0.535	+1.244	+0.976	+0.237	+0.597	+1.780	+1.518
Qwen3.5-9B	+0.219	+0.714	+0.892	+0.667	+0.233	+0.472	+1.074	+0.865	+0.221	+0.518	+1.554	+1.285
Open-Source Average	+0.146	+0.757	+0.850	+0.574	+0.143	+0.504	+1.083	+0.764	+0.147	+0.539	+1.568	+1.233
Overall Average	**+0.101**	**+0.745**	**+0.741**	**+0.480**	**+0.099**	**+0.498**	**+0.987**	**+0.678**	**+0.106**	**+0.528**	**+1.399**	**+1.066**

The regression results are consistent with the main analysis. Code Token Increment yields the strongest and most consistently significant negative coefficients, with most entries reaching the $p < 0.001$ level. Instruction Length and Annotation Type Count likewise show clear negative coefficients, with the largest magnitudes on Semantic Consistency and Design Effectiveness. Annotation Type Count has stronger effects than Annotation Count, suggesting that the variety of annotation forms is harder than the number of annotation elements alone. Visual Element Occupancy shows weaker and less stable effects. The Execution Rate coefficients are small in absolute terms, reflecting that execution failures are less sensitive to complexity than semantic and design quality. Open-source models also receive systematically larger negative coefficients than proprietary models across most indicators.

Table 20: Complex-vs-simple coefficients for Code Token Increment.

Model	**Exec.**	**Struct.**	**Sem.**	**Design**
Proprietary models				
GPT-5.4	-0.002	-0.068^{***}	-1.096^{***}	-0.995^{***}
Gemini 3.1 Pro Preview	-0.005^{*}	-0.075^{***}	-1.022^{***}	-0.869^{***}
Gemini 3 Flash Preview	-0.010^{**}	-0.147^{***}	-1.018^{***}	-0.860^{***}
Claude Sonnet 4.6	-0.024^{***}	-0.085^{***}	-1.140^{***}	-1.022^{***}
Open-source models				
Kimi K2.5	-0.030^{***}	-0.105^{***}	-1.297^{***}	-1.198^{***}
Gemma 4 31B	-0.064^{***}	-0.163^{***}	-1.466^{***}	-1.388^{***}
Qwen3.5-397B-A17B	-0.036^{***}	-0.126^{***}	-1.349^{***}	-1.302^{***}
Qwen3.5-122B-A10B	-0.078^{***}	-0.153^{***}	-1.481^{***}	-1.470^{***}
Qwen3.5-27B	-0.075^{***}	-0.157^{***}	-1.483^{***}	-1.455^{***}
Qwen3.5-9B	-0.141^{***}	-0.221^{***}	-1.626^{***}	-1.667^{***}

Table 21: Complex-vs-simple coefficients for Instruction Length.

Model	Exec.	Struct.	Sem.	Design
Proprietary models				
GPT-5.4	-0.002	-0.053***	-0.834***	-0.864***
Gemini 3.1 Pro Preview	-0.005	-0.049***	-0.701***	-0.684***
Gemini 3 Flash Preview	-0.008*	-0.131***	-0.744***	-0.724***
Claude Sonnet 4.6	-0.008	-0.045***	-0.834***	-0.842***
Open-source models				
Kimi K2.5	-0.023***	-0.073***	-0.962***	-1.035***
Gemma 4 31B	-0.036***	-0.111***	-1.027***	-1.137***
Qwen3.5-397B-A17B	-0.028***	-0.080***	-1.019***	-1.145***
Qwen3.5-122B-A10B	-0.071***	-0.113***	-1.173***	-1.346***
Qwen3.5-27B	-0.059***	-0.111***	-1.152***	-1.293***
Qwen3.5-9B	-0.113***	-0.152***	-1.319***	-1.481***

Table 22: Complex-vs-simple coefficients for Annotation Type Count.

Model	Exec.	Struct.	Sem.	Design
Proprietary models				
GPT-5.4	-0.002	-0.066***	-0.870***	-0.692***
Gemini 3.1 Pro Preview	-0.005	-0.058***	-0.847***	-0.641***
Gemini 3 Flash Preview	-0.019***	-0.111***	-0.936***	-0.729***
Claude Sonnet 4.6	-0.017**	-0.057***	-0.918***	-0.729***
Open-source models				
Kimi K2.5	-0.040***	-0.101***	-1.031***	-0.905***
Gemma 4 31B	-0.066***	-0.139***	-1.147***	-1.004***
Qwen3.5-397B-A17B	-0.076***	-0.158***	-1.231***	-1.065***
Qwen3.5-122B-A10B	-0.103***	-0.151***	-1.270***	-1.190***
Qwen3.5-27B	-0.138***	-0.182***	-1.339***	-1.233***
Qwen3.5-9B	-0.149***	-0.201***	-1.223***	-1.244***

Table 23: Complex-vs-simple coefficients for Annotation Count.

Model	Exec.	Struct.	Sem.	Design
Proprietary models				
GPT-5.4	-0.001	-0.008	-0.403***	-0.508***
Gemini 3.1 Pro Preview	-0.004	-0.030***	-0.357***	-0.428***
Gemini 3 Flash Preview	-0.006	-0.073***	-0.402***	-0.468***
Claude Sonnet 4.6	-0.007	-0.025***	-0.397***	-0.443***
Open-source models				
Kimi K2.5	-0.013*	-0.041***	-0.469***	-0.563***
Gemma 4 31B	-0.013	-0.070***	-0.462***	-0.592***
Qwen3.5-397B-A17B	-0.032***	-0.056***	-0.497***	-0.658***
Qwen3.5-122B-A10B	-0.053***	-0.066***	-0.554***	-0.742***
Qwen3.5-27B	-0.040***	-0.068***	-0.525***	-0.660***
Qwen3.5-9B	-0.083***	-0.091***	-0.674***	-0.854***

Table 24: Complex-vs-simple coefficients for Annotation Spatial Distribution Entropy.

Model	Exec.	Struct.	Sem.	Design
Proprietary models				
GPT-5.4	-0.006*	-0.040***	-0.253***	-0.295***
Gemini 3.1 Pro Preview	-0.007**	-0.033***	-0.245***	-0.267***
Gemini 3 Flash Preview	-0.008*	-0.071***	-0.302***	-0.307***
Claude Sonnet 4.6	-0.001	-0.028***	-0.250***	-0.289***
Open-source models				
Kimi K2.5	-0.000	-0.043***	-0.261***	-0.284***
Gemma 4 31B	-0.013	-0.058***	-0.298***	-0.349***
Qwen3.5-397B-A17B	-0.024***	-0.063***	-0.330***	-0.409***
Qwen3.5-122B-A10B	-0.036***	-0.073***	-0.367***	-0.490***
Qwen3.5-27B	-0.024**	-0.059***	-0.371***	-0.407***
Qwen3.5-9B	-0.042***	-0.081***	-0.417***	-0.521***

Table 25: Complex-vs-simple coefficients for Visual Element Occupancy.

Model	Exec.	Struct.	Sem.	Design
Proprietary models				
GPT-5.4	0.005	-0.020*	-0.220***	-0.227***
Gemini 3.1 Pro Preview	-0.001	-0.001	-0.275***	-0.304***
Gemini 3 Flash Preview	0.001	-0.011	-0.219***	-0.254***
Claude Sonnet 4.6	0.005	-0.000	-0.216***	-0.230***
Open-source models				
Kimi K2.5	-0.004	-0.011	-0.271***	-0.298***
Gemma 4 31B	-0.023*	-0.039***	-0.383***	-0.397***
Qwen3.5-397B-A17B	0.035***	0.013	-0.174**	-0.214***
Qwen3.5-122B-A10B	0.021	-0.009	-0.269***	-0.295***
Qwen3.5-27B	0.017	-0.001	-0.189**	-0.235***
Qwen3.5-9B	0.028	-0.006	-0.197**	-0.188**

J.3 Detailed Complexity Heatmaps

We further provide heatmaps grouped by Code Token Increment, the strongest complexity indicator in the main analysis. Figures 21, 22, and 23 show Structural Compliance, Semantic Consistency, and Design Effectiveness. Each heatmap reports model performance across simple, medium, and complex groups under three instruction levels and two input settings.

High-complexity cases remain difficult even under Implementation-level instructions, showing that concrete instructions do not fully remove the difficulty of complex annotation code.

K Runtime Error and Chart Fidelity Violation Statistics

Tables 26 and 27 report runtime error and chart fidelity violation distributions for all models. Percentages are computed within each model over the corresponding failure cases. In Tab. 26, Attr., Value, Type, Syntax, Name, Other, and Index denote AttributeError, ValueError, TypeError, SyntaxError, NameError, uncategorized runtime errors, and IndexError. In Tab. 27, Layout, Geometry, and Data Mark denote layout, figure geometry, and data mark violations.



Figure 21: Structural Compliance across Code Token Increment groups.



Figure 22: Semantic Consistency across Code Token Increment groups.



Figure 23: Design Effectiveness across Code Token Increment groups.

Table 26: Runtime error distributions by model. Total denotes the number of failed execution cases.

Model	Attr.	Value	Type	Syntax	Name	Other	Index	Total
Proprietary Models								
GPT-5.4	24.62%	35.38%	23.08%	10.77%	1.54%	3.08%	1.54%	65
Gemini 3.1 Pro Preview	38.46%	7.69%	25.64%	15.38%	12.82%	0.00%	0.00%	39
Gemini 3 Flash Preview	28.17%	23.94%	18.31%	14.08%	2.82%	12.68%	0.00%	71
Claude Sonnet 4.6	43.67%	8.86%	20.89%	15.82%	1.90%	6.33%	2.53%	158
Open-Source Models								
Kimi K2.5	25.36%	15.79%	16.75%	18.18%	15.31%	4.78%	3.83%	209
Gemma 4 31B	37.53%	23.11%	25.17%	8.92%	0.23%	4.35%	0.69%	437
Qwen3.5-397B-A17B	33.86%	21.75%	18.83%	9.64%	4.26%	8.74%	2.91%	446
Qwen3.5-122B-A10B	40.37%	24.87%	16.18%	6.30%	3.58%	4.94%	3.75%	587
Qwen3.5-27B	32.21%	31.24%	18.52%	4.03%	1.77%	7.57%	4.67%	621
Qwen3.5-9B	34.60%	22.34%	17.38%	9.46%	3.34%	4.97%	7.91%	1289

Table 27: Chart fidelity violation distributions by model. Total denotes the number of chart fidelity violation cases.

Model	Layout	Geometry	Data Mark	Total
Proprietary Models				
GPT-5.4	58.91%	1.18%	39.91%	679
Gemini 3.1 Pro Preview	78.35%	3.87%	17.78%	388
Gemini 3 Flash Preview	32.34%	52.86%	14.81%	1506
Claude Sonnet 4.6	62.90%	20.04%	17.06%	469
Open-Source Models				
Kimi K2.5	73.47%	12.60%	13.93%	524
Gemma 4 31B	56.52%	26.85%	16.62%	782
Qwen3.5-397B-A17B	70.05%	12.25%	17.70%	661
Qwen3.5-122B-A10B	75.00%	7.19%	17.81%	612
Qwen3.5-27B	67.07%	13.59%	19.34%	574
Qwen3.5-9B	70.79%	6.59%	22.61%	743

L Representative Error Cases

This section provides additional error examples for both rule-based and LLM-judged evaluation. We first show cases for *Chart Fidelity*, *Annotation Matching*, and *Color Matching*. We then show cases for *Semantic Faithfulness*, *Semantic Clarity*, *Visual Clarity*, *Annotation Organization Quality*, and *Attention Guidance*. The corresponding figures are collected in Sec. P.

L.1 Rule-based Evaluation Cases

Chart Fidelity. Fig. 25 shows a chart fidelity failure in an area chart task. The high-scoring result preserves the base chart structure. The low-scoring result is executable, but changes the plotting area and area proportions, which disrupts the base coordinate system and layout.

Annotation Matching. Fig. 26 shows an annotation matching failure in a line chart task. The low-scoring result partly follows the x-axis intent, but differs from the reference annotations in target objects, annotation count, and spatial positions.

Color Matching. Fig. 27 shows a color matching failure in a bar chart task. The reference chart uses orange to highlight only a small set of target bars, while the remaining bars keep the base teal color. The high-scoring result preserves this contrast. The low-scoring result adds annotation text and guide lines, but changes almost all bars to orange, making the target bars hard to distinguish from ordinary bars.

L.2 LLM-judged Evaluation Cases

Semantic Faithfulness. Fig. 28 shows a semantic faithfulness failure. The instruction asks the model to draw two boundary lines separating the outer summer region from the central winter region. The high-scoring result places the boundaries and region labels consistently with

the reference chart. The low-scoring result draws boundary lines, but reverses the summer and winter meanings.

Semantic Clarity. Fig. 29 shows a semantic clarity failure in a line chart task. The instruction asks the model to mark the price point corresponding to the Brexit referendum on June 23, 2016. The high-scoring result anchors the arrow and red marker to the correct point on the line. The low-scoring result has correct text and a clean layout, but the red marker floats above the curve, making the relation between the event and the data point unclear.

Visual Clarity. Fig. 30 shows a visual clarity failure in an event annotation task. The high-scoring result distributes event labels around the line chart and keeps the reading path clear. The low-scoring result stacks multiple dates, labels, and arrows in a small region, causing severe overlap. Although part of the intended content is present, the annotation text is difficult to read.

Annotation Organization Quality. Fig. 31 shows an annotation organization failure. The high-scoring result organizes the subtitle, explanatory annotation, and final data point with a clear visual hierarchy. The low-scoring result preserves the main content, but the subtitle moves downward and competes with the legend and source area. The explanatory annotation is also too long and extends beyond the chart area.

Attention Guidance. Fig. 32 shows an attention guidance failure in a bar chart task. The instruction aims to guide attention to the peak snowfall period from 6 p.m. to 11 p.m. The high-scoring result highlights the target interval with darker bars. The low-scoring result does not highlight the peak bars clearly, and its labels are far from the key region. Although the base chart remains readable, the intended focus is weak.

M Human Evaluation and LLM-Judge Validation Details

For each sample, raters were shown the annotation instruction, the unannotated base chart, the generated annotated chart, and the reference annotated chart. They independently rated each output using the same five 0–5 criteria used by the LLM judge: *Semantic Faithfulness*, *Semantic Clarity*, *Visual Clarity*, *Annotation Organization Quality*, and *Attention Guidance*.

A screenshot of the rating interface is shown in Fig. 24.

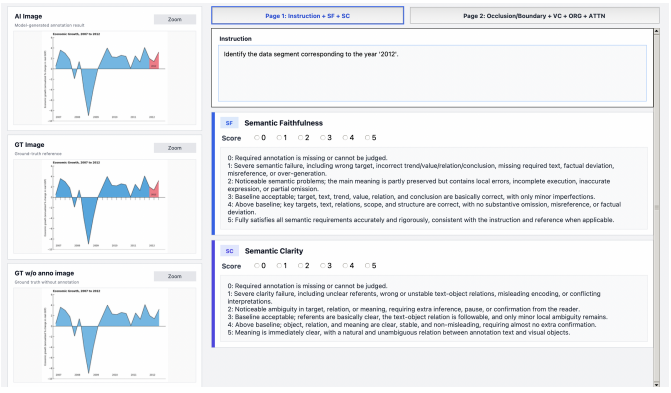


Figure 24: Screenshot of the human rating interface used for judge validation.

We provide additional validation of the LLM-based judge from two perspectives: (1) alignment with human ratings across outputs from models with different capability levels, and (2) potential judge-model bias through cross-judge consistency analysis.

M.1 Human Alignment Across Model Capability Levels

Following the human evaluation protocol described above, we validate whether GPT-5.4 judgments align with expert ratings across outputs from different model capability levels. We compute Spearman correlation between aggregated human ratings and averaged GPT-5.4 scores to measure human alignment. GPT-5.4 evaluates each sample three times, and ICC(3,1) across repeated judgments is used to measure judging stability. Tab. 31 reports the results.

Table 28: GPT-5.4 judge validation across output models. Spearman ρ measures alignment between averaged GPT-5.4 scores and aggregated human ratings. ICC(3,1) measures stability across three repeated GPT-5.4 runs.

		Human Alignment: Spearman ρ			Repeated-Run Stability: ICC(3,1)		
Output Model	N	**Semantic**	**Design**	**Overall**	**Semantic**	**Design**	**Overall**
Gemini 3.1 Pro Preview	300	0.7427	0.7341	0.7935	0.8615	0.9353	0.9156
Qwen3.5-9B	150	0.8027	0.8602	0.8678	0.9121	0.9453	0.9412
Qwen3.5-27B	150	0.7981	0.8326	0.8205	0.9005	0.9249	0.9281
Qwen Family	300	0.8097	0.8552	0.8513	0.9086	0.9378	0.9370
All Samples	600	**0.8194**	**0.8378**	**0.8593**	**0.9109**	**0.9474**	**0.9427**

Tab. 31 shows that alignment remains strong on outputs from weaker open-source models. For Qwen3.5-9B and Qwen3.5-27B outputs, the Overall Spearman correlations are 0.8678 and 0.8205, comparable to or higher than the 0.7935 on Gemini 3.1 Pro Preview outputs, with ICC(3,1) values of 0.9412 and 0.9281. This indicates that GPT-5.4 provides reliable judgments across different output-quality levels rather than only for outputs from a single strong model.

M.2 Human Alignment Across Instruction Levels

We further examine whether GPT-5.4 judgments remain consistent with human ratings across the three instruction levels. For each level, we compute Spearman correlation between aggregated human ratings and averaged GPT-5.4 scores, together with ICC(3,1) across the three repeated GPT-5.4 judgments. Tab. 32 reports the results.

Table 29: GPT-5.4 judge validation across instruction levels. Spearman ρ measures alignment between averaged GPT-5.4 scores and aggregated human ratings, while ICC(3,1) measures stability across three repeated GPT-5.4 runs. All Spearman correlations are significant at $p < .001$.

		Human Alignment: Spearman ρ		Repeated-Run Stability: ICC(3,1)		
Instruction Level	N	**Semantic**	**Design**	**Semantic**	**Design**	**Overall**
Intent	200	0.7912	0.8333	0.9113	0.9446	0.9441
Operation	200	0.8357	0.8362	0.9102	0.9327	0.9333
Implementation	200	0.7678	0.7312	0.8761	0.9444	0.9265

GPT-5.4 maintains strong human alignment and high repeated-run stability across all three instruction levels.

M.3 Cross-Judge Consistency and Judge-Model Bias

We compare three candidate judges: GPT-5.4, Claude Sonnet 4.6, and Gemini 3.1 Pro Preview. The judges evaluate outputs from five models: GPT-5.4, Claude Sonnet 4.6, Gemini 3.1 Pro Preview, Qwen3.5-27B, and Qwen3.5-9B. For each output model, we randomly sample 300 outputs, resulting in 1,500 evaluated outputs. All candidate judges use the same evaluation rubric. We measure pairwise agreement using Spearman correlation and overall consistency among judges using Cronbach's α.

Tab. 33 shows strong consistency among the three candidate judges. Pairwise Spearman correlations range from 0.546 to 0.937 across output models and metrics. Agreement is lowest on Semantic Consistency for GPT-5.4's own outputs (0.603–0.682), whereas the Overall-score agreement exceeds 0.86 for all five output models. As reported in the main paper, removing GPT-5.4 from the judge set leaves the ranking of the five output models unchanged. We provide additional validation of the LLM-based judge from two perspectives: (1) alignment with human ratings across outputs from models with different capability levels, and (2) potential judge-model bias through cross-judge consistency analysis.

Table 30: Cross-judge consistency among GPT-5.4, Claude Sonnet 4.6, and Gemini 3.1 Pro Preview. Pairwise agreement is measured using Spearman correlation, while Cronbach's α measures consistency among all three judges. Each output-model subset contains 300 samples.

Output Model	Metric	GPT–Claude ρ	GPT–Gemini ρ	Gemini–Claude ρ	Cronbach's α
GPT-5.4	Semantic Consistency	0.682	0.603	0.622	0.814
	Design Effectiveness	0.772	0.713	0.733	0.853
	Overall	0.760	0.697	0.731	0.865
Claude Sonnet 4.6	Semantic Consistency	0.744	0.649	0.669	0.852
	Design Effectiveness	0.759	0.715	0.734	0.877
	Overall	0.796	0.741	0.755	0.896
Gemini 3.1 Pro Preview	Semantic Consistency	0.726	0.569	0.546	0.805
	Design Effectiveness	0.752	0.684	0.695	0.844
	Overall	0.775	0.687	0.676	0.860
Qwen3.5-27B	Semantic Consistency	0.806	0.733	0.749	0.890
	Design Effectiveness	0.878	0.813	0.809	0.908
	Overall	0.875	0.803	0.817	0.918
Qwen3.5-9B	Semantic Consistency	0.861	0.813	0.825	0.927
	Design Effectiveness	0.898	0.830	0.840	0.921
	Overall	0.902	0.855	0.852	0.937
All Samples	Semantic Consistency	0.787	0.712	0.725	0.885
	Design Effectiveness	0.832	0.777	0.785	0.898
	Overall	0.842	0.784	0.793	0.913

M.4 Human Alignment Across Model Capability Levels

Following the human evaluation protocol described above, we validate whether GPT-5.4 judgments align with expert ratings across outputs from different model capability levels. We compute Spearman correlation between aggregated human ratings and averaged GPT-5.4 scores to measure human alignment. GPT-5.4 evaluates each sample three times, and ICC(3,1) across repeated judgments is used to measure judging stability. Tab. 31 reports the results.

Table 31: GPT-5.4 judge validation across output models. Spearman ρ measures alignment between averaged GPT-5.4 scores and aggregated human ratings. ICC(3,1) measures stability across three repeated GPT-5.4 runs.

		Human Alignment: Spearman ρ			Repeated-Run Stability: ICC(3,1)		
Output Model	N	Semantic	Design	Overall	Semantic	Design	Overall
Gemini 3.1 Pro Preview	300	0.7427	0.7341	0.7935	0.8615	0.9353	0.9156
Qwen3.5-9B	150	0.8027	0.8602	0.8678	0.9121	0.9453	0.9412
Qwen3.5-27B	150	0.7981	0.8326	0.8205	0.9005	0.9249	0.9281
Qwen Family	300	0.8097	0.8552	0.8513	0.9086	0.9378	0.9370
All Samples	600	**0.8194**	**0.8378**	**0.8593**	**0.9109**	**0.9474**	**0.9427**

Tab. 31 shows that alignment remains strong on outputs from weaker open-source models. For Qwen3.5-9B and Qwen3.5-27B outputs, the Overall Spearman correlations are 0.8678 and 0.8205, comparable to or higher than the 0.7935 on Gemini 3.1 Pro Preview outputs, with ICC(3,1) values of 0.9412 and 0.9281. This indicates that GPT-5.4 provides reliable judgments across different output-quality levels rather than only for outputs from a single strong model.

M.5 Human Alignment Across Instruction Levels

We further examine whether GPT-5.4 judgments remain consistent with human ratings across the three instruction levels. For each level, we compute Spearman correlation between aggregated human ratings and averaged GPT-5.4 scores, together with ICC(3,1) across the three repeated GPT-5.4 judgments. Tab. 32 reports the results.

Table 32: GPT-5.4 judge validation across instruction levels. Spearman ρ measures alignment between averaged GPT-5.4 scores and aggregated human ratings, while ICC(3,1) measures stability across three repeated GPT-5.4 runs. All Spearman correlations are significant at $p < .001$.

		Human Alignment: Spearman ρ		Repeated-Run Stability: ICC(3,1)		
Instruction Level	N	Semantic	Design	Semantic	Design	Overall
Intent	200	0.7912	0.8333	0.9113	0.9446	0.9441
Operation	200	0.8357	0.8362	0.9102	0.9327	0.9333
Implementation	200	0.7678	0.7312	0.8761	0.9444	0.9265

GPT-5.4 maintains strong human alignment and high repeated-run stability across all three instruction levels.

M.6 Cross-Judge Consistency and Judge-Model Bias

We compare three candidate judges: GPT-5.4, Claude Sonnet 4.6, and Gemini 3.1 Pro Preview. The judges evaluate outputs from five models: GPT-5.4, Claude Sonnet 4.6, Gemini 3.1 Pro Preview, Qwen3.5-27B, and Qwen3.5-9B. For each output model, we randomly sample 300 outputs, resulting in 1,500 evaluated outputs. All candidate judges use the same evaluation rubric. We measure pairwise agreement using Spearman correlation and overall consistency among judges using Cronbach's α.

Table 33: Cross-judge consistency among GPT-5.4, Claude Sonnet 4.6, and Gemini 3.1 Pro Preview. Pairwise agreement is measured using Spearman correlation, while Cronbach's α measures consistency among all three judges. Each output-model subset contains 300 samples.

Output Model	Metric	GPT–Claude ρ	GPT–Gemini ρ	Gemini–Claude ρ	Cronbach's α
GPT-5.4	Semantic Consistency	0.682	0.603	0.622	0.814
	Design Effectiveness	0.772	0.713	0.733	0.853
	Overall	0.760	0.697	0.731	0.865
Claude Sonnet 4.6	Semantic Consistency	0.744	0.649	0.669	0.852
	Design Effectiveness	0.759	0.715	0.734	0.877
	Overall	0.796	0.741	0.755	0.896
Gemini 3.1 Pro Preview	Semantic Consistency	0.726	0.569	0.546	0.805
	Design Effectiveness	0.752	0.684	0.695	0.844
	Overall	0.775	0.687	0.676	0.860
Qwen3.5-27B	Semantic Consistency	0.806	0.733	0.749	0.890
	Design Effectiveness	0.878	0.813	0.809	0.908
	Overall	0.875	0.803	0.817	0.918
Qwen3.5-9B	Semantic Consistency	0.861	0.813	0.825	0.927
	Design Effectiveness	0.898	0.830	0.840	0.921
	Overall	0.902	0.855	0.852	0.937
All Samples	Semantic Consistency	0.787	0.712	0.725	0.885
	Design Effectiveness	0.832	0.777	0.785	0.898
	Overall	0.842	0.784	0.793	0.913

Tab. 33 shows strong consistency among the three candidate judges. Pairwise Spearman correlations range from 0.546 to 0.937 across output models and metrics. Agreement is lowest on Semantic Consistency for GPT-5.4's own outputs (0.603–0.682), whereas the Overall-score agreement exceeds 0.86 for all five output models. As reported in the main paper, removing GPT-5.4 from the judge set leaves the ranking of the five output models unchanged.

N Detailed Results for Cross-Representation Generalization

Tab. 34 reports the complete model-level results for the D3 and SVG extensions discussed in Sec. 5 of the main paper. These results complement the aggregated comparison in Fig. 12 of the main paper and provide the performance of each evaluated model across all three instruction levels. These model-level results echo the two findings summarized in Sec. 5 of the main paper. First, the key trends established under Python carry over to the new representations: more specific instructions generally yield higher *Semantic Consistency* and *Design Effectiveness*, and the overall model ranking under D3 and SVG remains close to that under Python. Second, the two representations exhibit different trade-offs across instruction levels, with D3 leading on the rule-based metrics while SVG slightly overtakes D3 on the LLM-judged metrics at the Implementation level. As in the main evaluation, Intent-level *Structural Compliance* reports *Chart Fidelity* only and is therefore shown separately from the Structural Compliance scores at the Operation and Implementation levels.

O Use of AI Assistants

AI assistants were used to support initial chart reconstruction, annotation removal, structured annotation drafting, instruction drafting, coding, figure drafting, and language polishing. All outputs were manually checked and revised by the authors. The LLM-based judge used in our experiments is part of the proposed evaluation method and is described in Sec. 3.3 of the main paper and Sec. E of this supplementary document.

P Gallery of Representative Error Cases

Table 34: Complete model-level evaluation results under D3 (left) and SVG (right) representations. Metrics, notation, and highlighting conventions follow Tab. 3 of the main paper. Gray Struct.* columns report *Chart Fidelity* only for Intent-level instructions and are not directly comparable with *Structural Compliance* at the other levels.

Model	D3												SVG											
	Intent-level				Operation-level				Implementation-level				Intent-level				Operation-level				Implementation-level			
	Exec.	Struct.*	Sem.	Design	Exec.	Struct.	Sem.	Design	Exec.	Struct.	Sem.	Design	Exec.	Struct.*	Sem.	Design	Exec.	Struct.	Sem.	Design	Exec.	Struct.	Sem.	Design
Proprietary Models																								
GPT-5.4	**1.000**	**0.827**	2.929	3.125	**0.992**	**0.752**	3.008	3.258	0.983	0.771	3.471	3.555	**1.000**	**0.828**	2.738	3.106	**1.000**	0.691	2.767	3.175	**1.000**	0.743	3.621	3.706
Gemini 3.1 Pro Preview	0.983	0.809	**3.084**	**3.317**	0.983	0.747	**3.246**	**3.393**	0.992	**0.807**	**3.580**	**3.681**	**1.000**	0.817	**3.042**	**3.320**	**1.000**	**0.728**	**3.146**	**3.511**	**1.000**	**0.802**	**3.696**	**3.736**
Gemini 3 Flash Preview	0.992	0.783	3.017	3.078	0.983	0.737	3.122	3.359	0.967	0.804	3.517	3.635	0.983	0.737	2.917	3.267	**1.000**	0.693	2.871	3.333	0.992	0.778	3.667	3.694
Claude Sonnet 4.6	**1.000**	0.723	3.013	2.981	**0.992**	0.728	3.029	3.090	**1.000**	0.788	3.517	3.503	**1.000**	0.633	2.779	3.003	**1.000**	0.667	2.746	3.231	**1.000**	0.801	3.646	3.625
Open-Source Models																								
Kimi K2.5	**0.983**	0.773	**2.777**	**2.919**	**1.000**	**0.745**	**2.842**	**3.031**	**0.992**	**0.806**	**3.329**	**3.403**	**0.992**	**0.792**	**2.338**	**2.892**	**0.992**	**0.658**	**2.367**	**2.981**	**0.992**	**0.773**	**3.338**	3.517
Gemma 4 31B	0.958	0.824	2.722	2.829	0.975	0.675	2.551	2.858	0.958	0.716	3.217	3.328	0.842	0.720	2.091	2.625	0.833	0.510	2.072	2.683	0.817	0.635	3.103	3.360
Qwen3.5-397B-A17B	0.958	0.761	2.635	2.777	0.967	0.680	2.530	2.805	0.967	0.722	3.211	3.399	0.842	0.685	2.197	2.724	0.825	0.486	2.117	2.809	0.825	0.554	3.270	**3.562**
Qwen3.5-122B-A10B	0.950	**0.833**	2.496	2.667	0.975	0.701	2.500	2.735	**0.992**	0.746	3.029	3.177	0.792	0.683	2.126	2.596	0.792	0.572	2.055	2.667	0.800	0.659	3.165	3.423
Qwen3.5-27B	0.875	0.777	2.367	2.676	0.867	0.674	2.279	2.654	0.783	0.720	2.888	3.163	0.817	0.676	1.966	2.418	0.825	0.509	2.087	2.651	0.817	0.649	3.167	3.389
Qwen3.5-9B	0.875	0.753	2.010	2.333	0.875	0.705	1.967	2.406	0.850	0.736	2.392	2.716	0.733	0.644	1.442	2.153	0.733	0.499	1.446	2.226	0.800	0.637	2.657	3.061

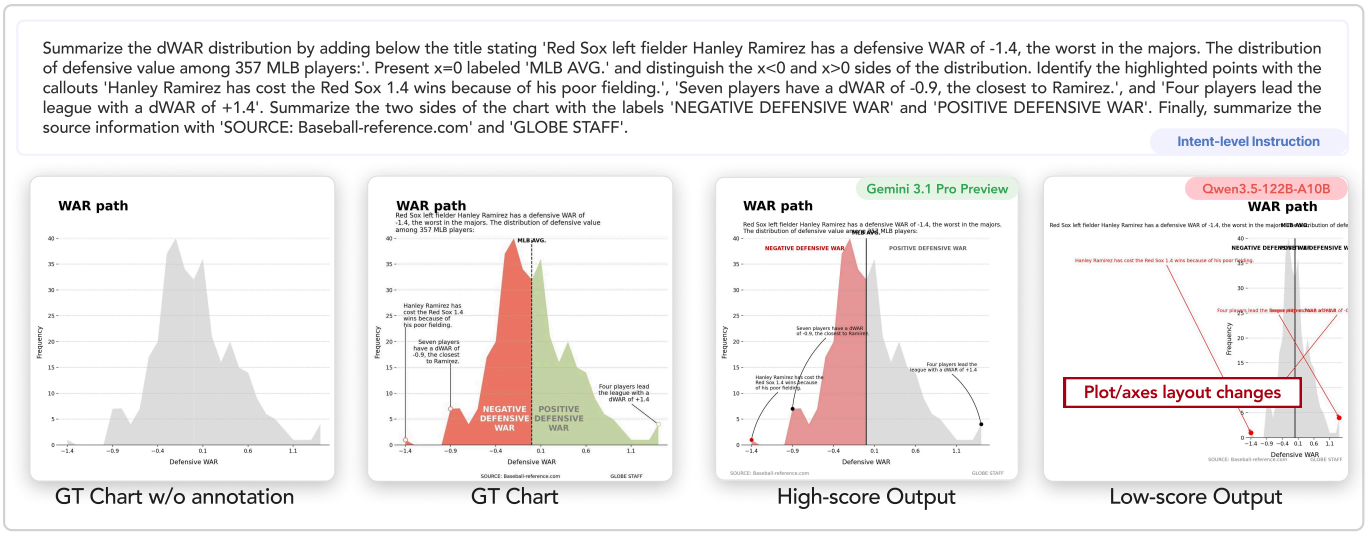


Figure 25: Representative error case for rule-based *Chart Fidelity*. The low-scoring and high-scoring outputs are produced under the same instruction and reference chart.

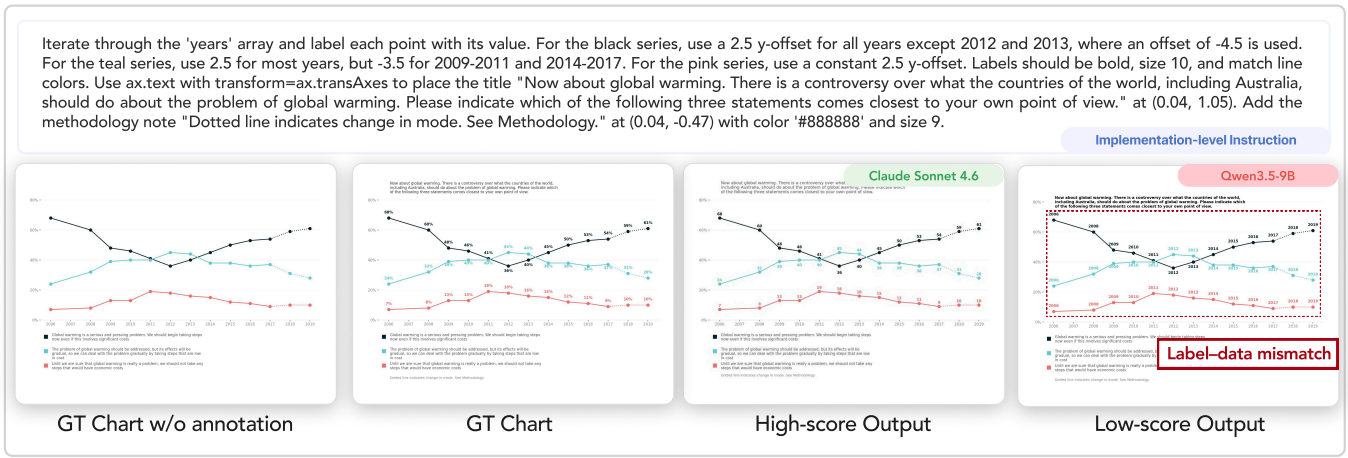


Figure 26: Representative error case for rule-based *Annotation Matching*. The low-scoring and high-scoring outputs are produced under the same instruction and reference chart.

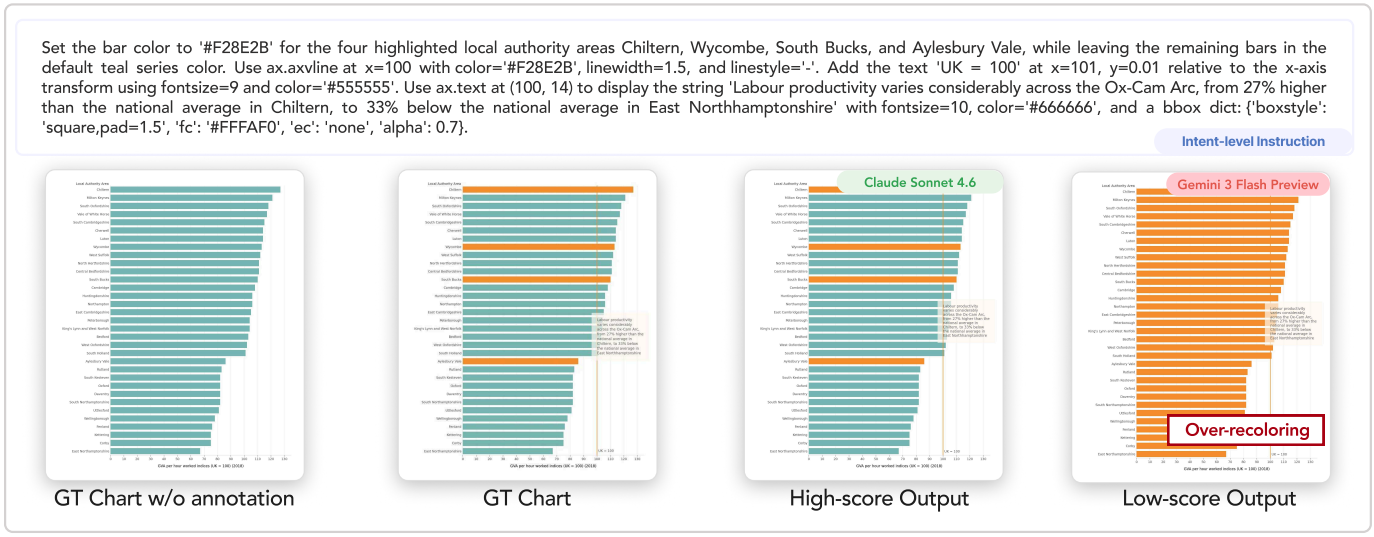


Figure 27: Representative error case for rule-based *Color Matching*. The low-scoring and high-scoring outputs are produced under the same instruction and reference chart.

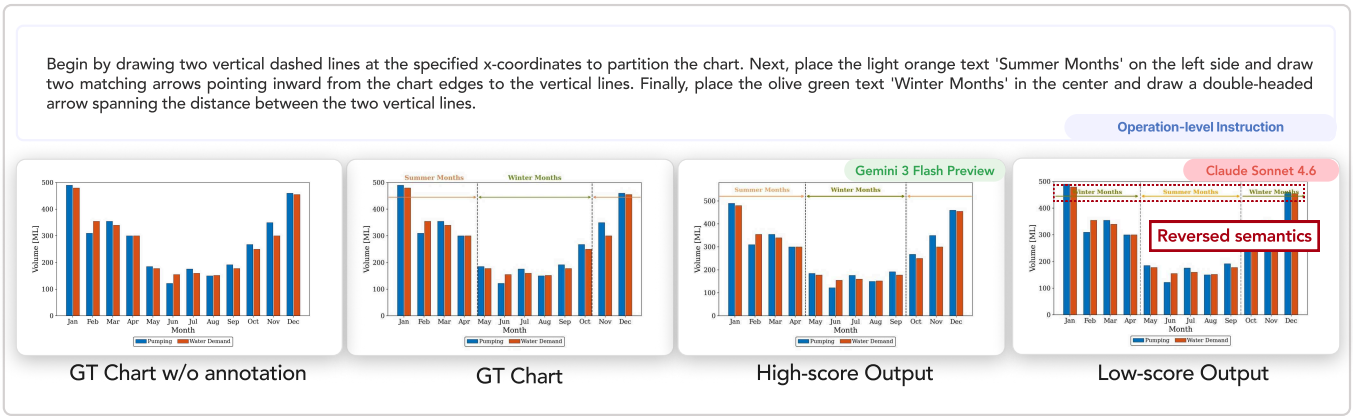


Figure 28: Representative error case for LLM-judged *Semantic Faithfulness*. The low-scoring and high-scoring outputs are produced under the same instruction and reference chart.

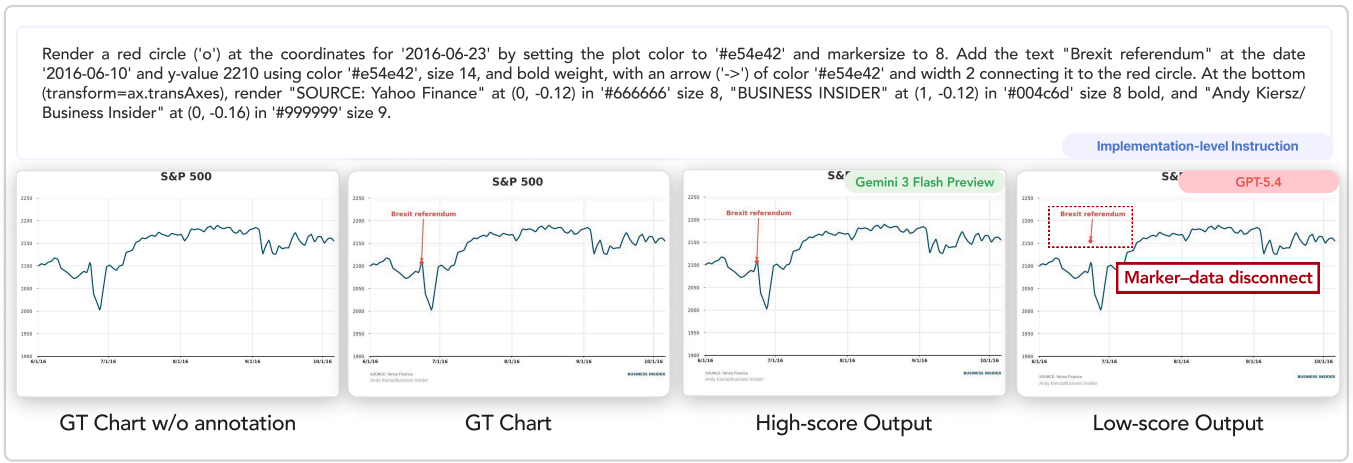


Figure 29: Representative error case for LLM-judged *Semantic Clarity*. The low-scoring and high-scoring outputs are produced under the same instruction and reference chart.

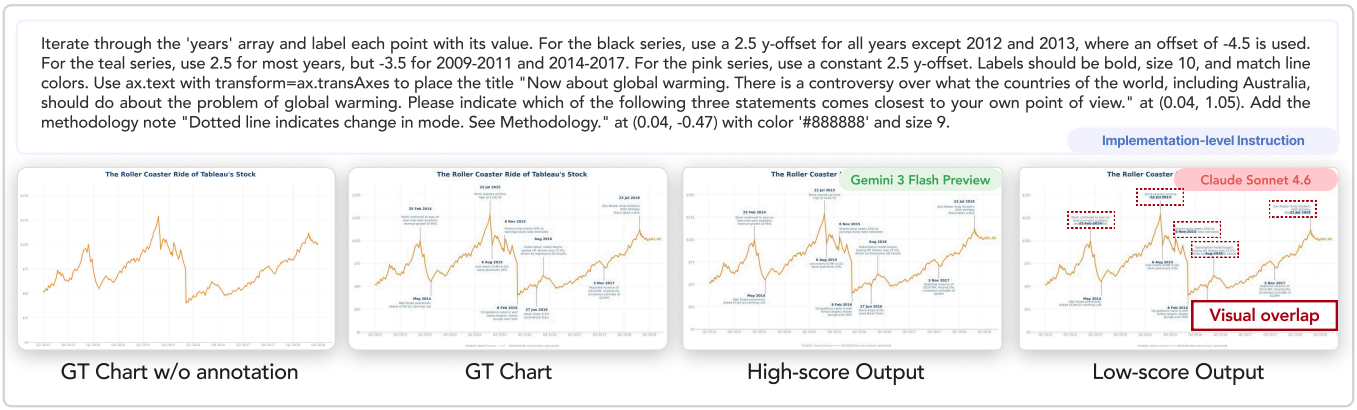


Figure 30: Representative error case for LLM-judged *Visual Clarity*. The low-scoring and high-scoring outputs are produced under the same instruction and reference chart.

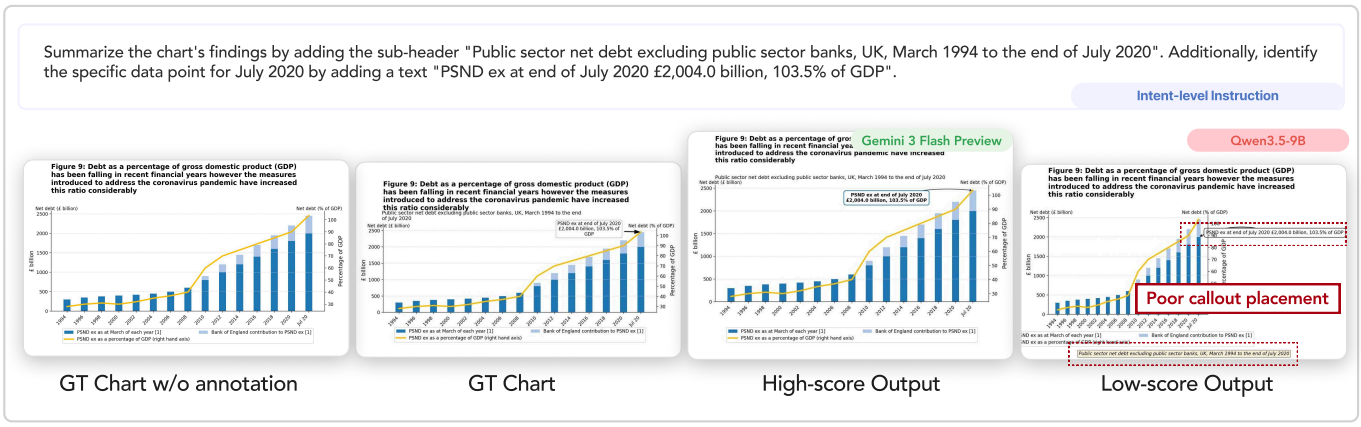


Figure 31: Representative error case for LLM-judged *Annotation Organization Quality*. The low-scoring and high-scoring outputs are produced under the same instruction and reference chart.

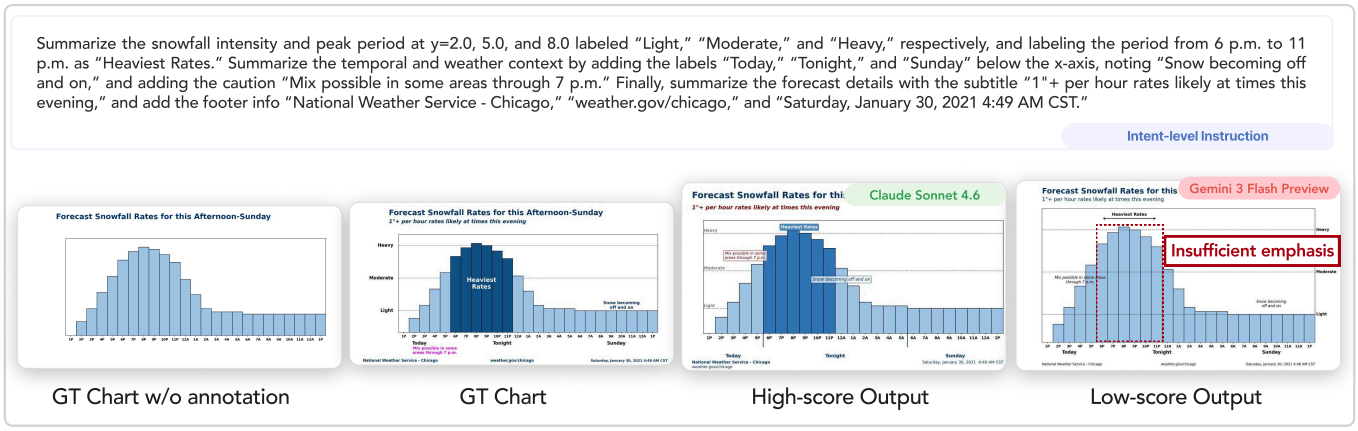


Figure 32: Representative error case for LLM-judged *Attention Guidance*. The low-scoring and high-scoring outputs are produced under the same instruction and reference chart.